UNIVERSITETET I OSLO
Institutt for informatikk

Digital Manusflyt

Ruth Merethe Evang

21. april 2005

Innhold

Figurer

Kapittel 1

Forord

Jeg hadde nettopp skrevet en liten barnebok da jeg skulle velge meg en masteroppgave. Da jeg så at J.W. Cappelens Forlag AS ønsket en masterstudent til å arbeide med systemet deres, grep jeg sjansen for å lære mer om hvordan bokproduksjon fungerer. Prosjektet utviklet seg spennende, og jeg fattet etter hvert stor interesse for problemområdet.

Jeg hadde planer om å skrive om et testprosjekt i tillegg, men på grunn av liten tid for forlaget ble dette utsatt til etter at oppgaven min skulle være levert. Dette førte til at oppgaven inneholder få resultater, og jeg fokuserte mer på den opprinnelige problemstillingen.

Det har vært et morsomt prosjekt, og jeg har fått samarbeide med mange hyggelige personer. Jeg vil takke Peter Hausken, Paul Heisholt, Gisle Hannemyr, Harriet Karoliussen, Randi Faye, Håkon Wium Lie, Frank Olaf Sem-Jacobsen, Roar Granerud og Turid Evang for hjelp og samarbeid under arbeidet med masteroppgaven. I tillegg vil jeg takke firmaene J.W. Cappelens Forlag, InfoFuture og PDC Tangen.

Kapittel 2

Begreper

I dette kapittelet introduserer vi en del begreper som inngår i resten av oppgaven.

2.1 Semantikk

Vi kan dele språket inn i de tre forskjellige kategoriene syntaks, grammatikk og semantikk. Syntaks er bruk av tegn, og hvordan de forskjellige bokstavene i ordene skal brukes. Grammatikk er hvordan setninger skal bygges opp av de forskjellige ordene, og hvordan ordene skal bøyes. Semantikk er betydningen av ordene og setningene. Disse begrepene kan overføres til dataverdenen.

Når begrepet semantikk brukes om markeringsspråk mener vi hvilken rolle teksten spiller i den logiske strukturen. En tekst kan være bygget opp av mange forskjellige elementer som har ulik rolle. La oss si at vi har en kokebok. Den inneholder en stor mengde med tekst. Her ser vi hvordan teksten til en av oppskriftene i kokeboka kan se ut:

```
Ostesmørbrød

Med mais, sjampingjong, basilikum og ost.
5 min

Du trenger:

    * Så mange skiver som du spiser selv
    * Hermetisk mais
    * Hermetisk sjampingjong
    * Basilikum (helst fersk)
    * Salt eller trocomare
    * Gulost (jeg synes gräddost er best)
```

```
Slik gjør du:

Legg mais og sjampingjong på brødskivene. Ha gulost øverst med
basilikum og salt på. Stekes i ovnen på ca 180 grader til osten
har blitt sprø. Kan også stekes i microbølgeovn, men resultatet
blir ikke like bra. Da stekes ostesmørbrødene til osten har
smeltet.
```

Et menneske kan se hva som er navnet på retten, og hvilke ingredienser som skal brukes. Men for en datamaskin er dette kun en rekke med tilfeldige tegn. For at maskinen skal få bedre forståelse for innholdet må man eksplisitt forklare hva som er hva. Ved hjelp av for eksempel språket XML (Vi tar for oss XML i seksjon 5.3 på side 38) kan vi gjøre dette.

```
<oppskrift>
    <navn>Ostesmørbrød</navn>
    <beskrivelse>Med mais, sjampingjong, basilikum og ost.</beskrivelse>
    <tid>5 min</tid>

    <ingredienser>
        <ingrediens>Så mange skiver som du spiser selv</ingrediens>
        <ingrediens>Hermetisk mais</ingrediens>
        <ingrediens>Hermetisk sjampingjong</ingrediens>
        <ingrediens>Basilikum (helst fersk)</ingrediens>
        <ingrediens>Salt eller trocomare</ingrediens>
        <ingrediens>Gulost (jeg synes gräddost er best)</ingrediens>
    </ingredienser>

    <utførelse>
    Legg mais og sjampingjong på brødskivene. Ha gulost øverst med
    basilikum og salt på. Stekes i ovnen på ca 180 grader til osten
    har blitt sprø. Kan også stekes i microbølgeovn, men resultatet
    blir ikke like bra. Da stekes ostesmørbrødene til osten har
    smeltet.
    </utførelse>
</oppskrift>
```

Her forklarer vi hva hvert element er til datamaskinen. Inne i klammene står det en kort beskrivelse som vi kan ha definert på forhånd, slik at datamaskinen forstår hvordan det som står mellom startklammen (for eksempel <navn>) og sluttklammen (for eksempel </navn>) skal behandles.

Ostesmørbrød

Med mais, sjampingjong, basilikum og ost.
5 min

Du trenger:

* Så mange skiver som du spiser selv
* Hermetisk mais
* Hermetisk sjampingjong
* Basilikum (helst fersk)
* Salt eller trocomare
* Gulost (jeg synes gräddost er best)

Slik gjør du:

Legg mais og sjampingjong på brødskivene. Ha gulost øverst med basilikum og salt på. Stekes i ovnen på ca 180 grader til osten har blitt sprø. Kan også stekes i microbølgeovn, men resultatet blir ikke like bra. Da stekes ostesmørbrødene til osten har smeltet.

Figur 2.1: Eksempel på format som tar vare på den grafiske utformingen

Når vi snakker om et dokument som er semantisk strukturert, mener vi at lagringsformatet tar vare på den semantiske informasjonen. Eksempelet viser hvordan dette kan gjøres med XML. Når jeg snakker om såkalte "enkle" bøker videre i oppgaven mener jeg bøker som har en enkel struktur, som for eksempel romaner.

2.2 Grafisk utforming

En annen måte å lagre dokumenter på er å bruke et format som tar vare på informasjon om hvordan et dokument skal se ut. Under ser vi et eksempel på et dokument laget i WordPad. Den bruker RTF-standarden (Mer om dette i seksjon 5.2.3 på side 36) som tar vare på den grafiske utformingen. Dette kan for eksempel være skrifttype og skriftstørrelse.

Hvis vi ser på dette i en tekstbehandler, kan vi se hvordan utseendeinformasjonen er lagret.

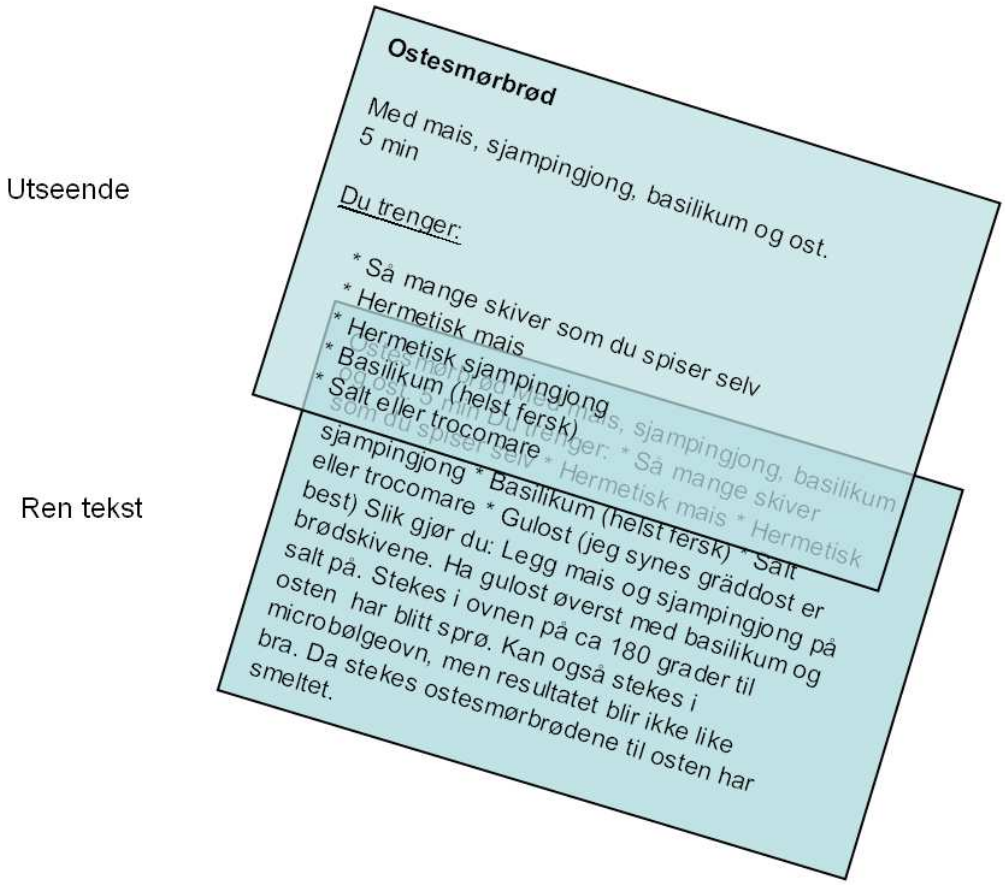
```
{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fscript\fprq2
```

```
\fcharset0 Comic Sans MS;}{\f1\fswiss\fcharset0 Arial;}}
{\*\generator Msftedit 5.41.15.1507;}\viewkind4\uc1\pard\b\f0\fs28
Ostesm\'f8rbr\'f8d\par
\b0\fs20\par
\i Med mais, sjampingjong, basilikum og ost.\par
5 min\par
\i0\par
\ul Du trenger:\par
\ulnone\par
    * S\'e5 mange skiver som du spiser selv\par
    * Hermetisk mais\par
    * Hermetisk sjampingjong\par
    * Basilikum (helst fersk)\par
    * Salt eller trocomare\par
    * Gulost (jeg synes gr\'e4ddost er best)\par
\par
\ul Slik gj\'f8r du:\par
\ulnone\par
Legg mais og sjampingjong p\'e5 br\'f8dskivene. Ha gulost \'f8verst med \par
basilikum og salt p\'e5. Stekes i ovnen p\'e5 ca 180 grader til osten \par
har blitt spr\'f8. Kan ogs\'e5 stekes i microb\'f8lgeovn, men resultatet \par
blir ikke like bra. Da stekes ostesm\'f8rbr\'f8dene til osten har \par
smeltet.\f1\par
}
```

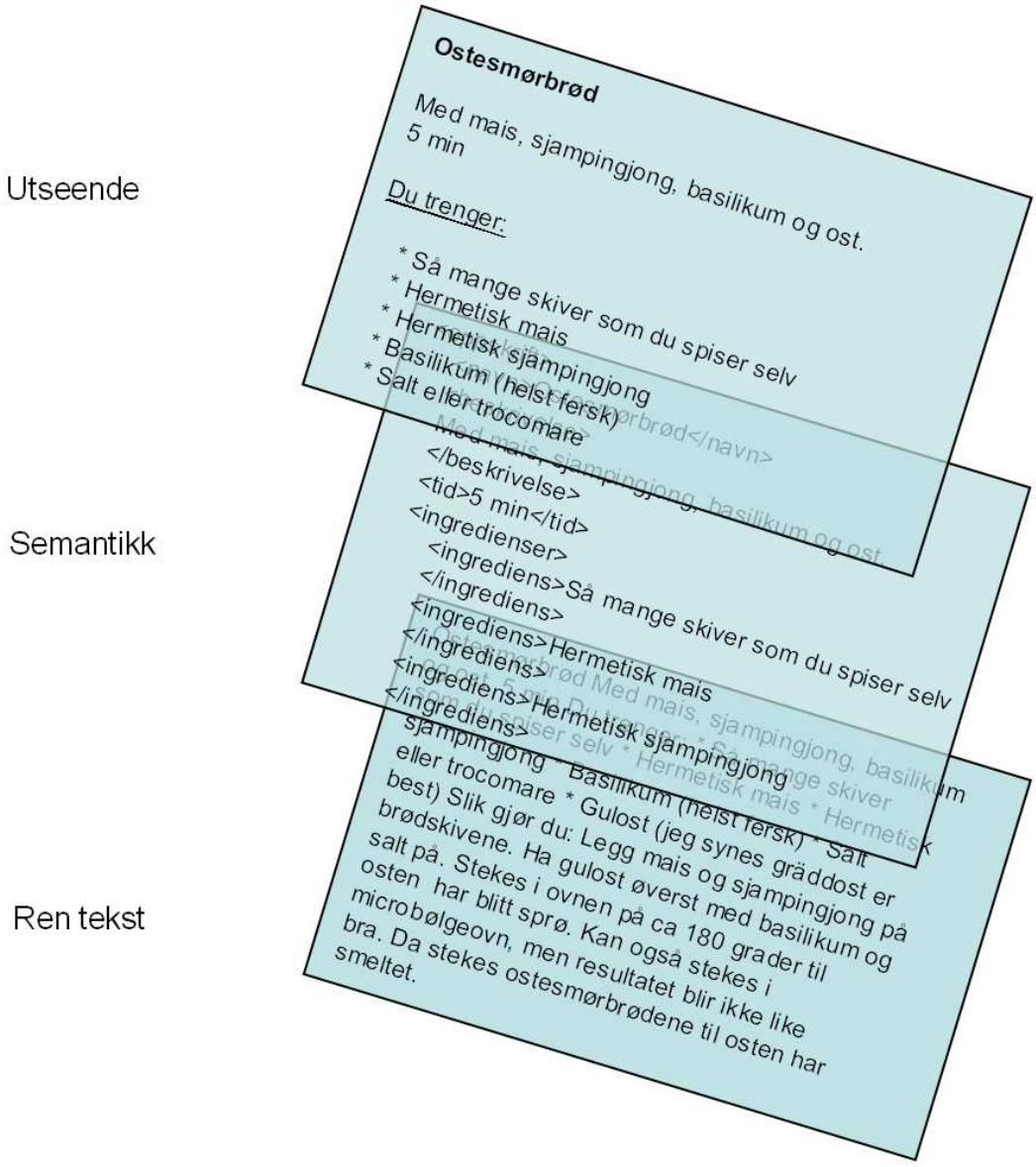
Dette kan se ganske uoversiktelig ut, og kanskje litt kryptisk. Spesielt fordi det leses med et kodesett som ikke inneholder æ, ø og å, og disse blir erstattet med andre tegn. Men vi kan for eksempel se at vi har brukt skrifttypen 'Comic Sans MS', og hvor vi har satt linjeskift (par).

De forskjellige formatene kan ta vare på forskjellig informasjon. Enkelte formater tar ikke vare på grafisk utforming i det hele tatt, og man trenger eksterne stilark for fremvisning. Enkelte andre formater tar for eksempel ikke vare på semantikk. Dette er ikke noe man trenger å ta vare på når man kun skal fremvise noe, så formater kan klare seg uten det semantiske laget. Vi har også formater som både tar vare på teksten, semantikken og den grafiske utformingen. Figur 2.2 på neste side viser et format som inneholder ren tekst og informasjon om den grafiske utformingen. Her ser vi at det ligger et lag med grafisk utformingsinformasjon over den rene teksten. Mens figur 2.3 viser et format som i tillegg har med et semantisk lag.

Man kan konvertere fra et format som inneholder mer informasjon enn det formatet som skal konverteres til. Man kan for eksempel konvertere fra et format som tar vare på både semantikk og grafisk



Figur 2.2: Grafisk utforming og tekstlagene.



Figur 2.3: Grafisk utforming, semantikk- og tekstlagene.

utforming til et format som kun inneholder grafisk utforming i tillegg til den rene teksten.

2.3 Godseid format

At et format er godseid (eng. proprietary) betyr at det er eiet av en gruppe bedrifter eller personer hvor medlemsskapet er lukket. Med dette mener vi at det er en bedrift som kan bestemme hva andre har lov til å gjøre med formatet. Om det for eksempel kan brukes fritt, at dokumentene kan distribueres fritt, eller at de kan endres fritt. Bedriftene kan ha en egen lisens for de forskjellige godseide formatene sine. Et godseid format er beskyttet av opphavsrett.

2.4 Fritt format

At et format er fritt betyr at det kontrolleres av en åpen organisasjon, brukergruppe eller konsortium der medlemsskapet er åpent. Det vil si at man har mulighet til å bli med på å kontrollere formatet hvis man ønsker dette. HTML er et eksempel på et fritt format. Standarden blir kontrollert gjennom den åpne organisasjonen W3C.

2.5 Åpent format

Definisjonen på at et format er åpent, er at all dokumentasjonen til formatet er offentlig tilgjengelig. Dette betyr ikke at man har noen andre former for rettigheter. Et format kan altså være både godseid og åpent.

2.6 Lukket format

Et lukket format er et format der dokumentasjonen ikke er offentlig tilgjengelig.

Figur 2.4 på neste side er en oversikt over kombinasjoner av fritt/godseid format, og åpent/lukket format. Det er vanskelig å tenke seg et format som både er fritt og lukket. Når dokumentasjonen ikke er tilgjengelig kan ikke formatet være fritt heller. Men de andre tre kombinasjonene finnes.

	Fritt	Godseid
Åpent	Formatet kontrolleres av en åpen organisasjon.	Formatet er eid av en gruppe, men dokumentasjonen er offentlig tilgjengelig.
Lukket		Formatet er eid av en gruppe, og dokumentasjonen er ikke offentlig tilgjengelig

Figur 2.4: Kombinasjoner av fritt/godseid og åpent/lukket

Kapittel 3

Innledning

Da man innførte bruk av datamaskinen som hjelpemiddel i bokbransjen ble det mer effektivt å produsere bøker. Datamaskiner blir i dag stort sett brukt som avanserte skrivemaskiner hos bokbransjen, og man utnytter ikke potensialet til digitalt lagrede manus. Mennesker er ofte resultatorienterte, og tenker ikke over hvilke muligheter man har hvis man legger ned litt mer arbeid med et digitalt manus.

Det kan være enklest å konsentrere seg om hvordan boka vil se ut til slutt, og det er få som har gjort omfattende studier innenfor strukturert lagring. Hvis vi kan finne en bra måte å ta vare på semantikken i bøker åpner det for nye muligheter innenfor bokbransjen og andre steder man har digitalt lagret informasjon.

Det finnes forskjellige nivåer av semantisk strukturering når vi snakker om en bok. Det første nivået er å strukturere basiselementene som den består av. Dette kan for eksempel være kapitler og avsnitt.

Hvis man går enda lengre kan man strukturere en dypere mening for innholdet. Et eksempel på dette kan være å strukturere kjente personer eller hendelser fra historiebøker. Ved å kode semantisk vil man kanskje kunne trekke ut en tidslinje om en bestemt persons liv, eller et steds historie. Dette er tanker som ligger et godt stykke fram i tid, men ved hjelp av et strukturert lagringsformat vil det være mulig å gjennomføre.

Andre bruksområder for strukturert lagring kan være hvis man ønsker å lenke til andre skrevne dokumenter. Det kan for eksempel være interessant å ha et sammendrag av teksten sin strukturert. På denne måten kan andre bøker, artikler eller linkende kan trekke ut og inkludere sammendraget i sin egen tekst.

Problemet er at man ikke har kommet fram til noen god måte å strukturere bøker på. Vi har kommet et godt sted på veien, ved at man nå kan få ordbøker og leksikon elektronisk. Men det er ikke funnet like greie måter å strukturere for eksempel romaner. Et probleme med romaner er at de ofte har en løs struktur. I denne oppgaven skal vi se på hvordan

man kan strukturere romaner. Etter hvert kan dette utvides til å dekke mer avansert oppbygde bøker.

3.1 Problemstilling

J.W. Cappelens Forlag er et norsk bokforlag som ønsker å oppdatere systemet sitt nå som nye muligheter innenfor strukturert, digital lagring har dukket opp. Forlaget er interessert i å komme fram til en mer effektiv bokproduksjon ved hjelp av ny teknologi.

Tidlig i 2003 ble datasystemet Sparta introdusert for hele bedriften. Dette var et stort steg på vei mot en mer effektiv arbeidsprosess for forlaget, ved at all data kunne lagres på samme sted og på samme måte. I denne oppgaven vil vi ta for oss en liten del av Cappelens arbeidsprosess, og prøve å komme fram til et forslag til en effektivisering av denne biten. Delen vi skal se på er arbeidet med et manus fra det er antatt av forlaget, til det sendes til trykk.

Cappelen får inn mange manus fra sine forskjellige forfattere hver uke. Hvis forlaget bestemmer seg for å utgi manuset sier vi at det er antatt. Etter at et manus er antatt må det redigeres manuelt av redaksjonen. All formattering og design blir lagt til manuelt. Dette er en tidkrevende prosess som Cappelens IT-avdeling mener kan gjøres på en enklere og bedre måte. (Se figur 4.1 for kart over nåværende manusflyt)

Deretter lagres hele dokumentet som et ferdig PDF-dokument i Cappelens system. Grafisk utforming og innhold blir derfor lagret sammen, og det blir en tidkrevende prosess å kunne endre manuset i ettertid. Dette er lite gunstig med tanke på gjenbruk, noe som er viktig for noen utgivelser av bøker.

Det første vi må se på er hvilke langringsformater eller standarder som er passende for lagring av bøker. Deretter må vi finne en måte å komme fra forfatterens skriverier, til lagring i formatet vi kommer fram til. Oppgaven vil altså dreie seg om å komme fram til en arbeidsmåte og en arbeidsfordeling mellom forfatter og redaksjon som oppnår dette på raskest og best mulig måte.

Denne oppgaven er begynnelsen på Cappelens igangsetting av et nytt system. Her ser vi på hvilke muligheter forlaget har, og presenterer et forslag til en løsning. Det neste steget er å sette igang et lite testprosjekt der forslaget blir testet ut.

3.2 J.W. Cappelens Forlag

Cappelen ble opprettet i 1829 av Jørgen Wright Cappelen, og er Norges eldste eksisterende forlag. Bonnier AB er det største forlaget i

Skandinavia, og har vært Cappelens eier siden 1987. Cappelens kontorer finnes i Oslo, og bedriften består av rundt 300 ansatte.

Forlaget publiserer både skjønnlitteratur, undervisningsbøker og faktabøker, og det publiserer i gjennomsnitt omkring 1000 titler hvert år. I tillegg driver Cappelen fire bokklubber, og eier en lagrings- og distribusjonsbedrift (Sentraldistribusjonen AS). I 2003 hadde bedriften et omsetning på over 600 millioner NOK.

Det totale bokmarkedet i Norge ligger på rundt 5 milliarder kroner. I 2002 ble hele 17 millioner bøker solgt i Norge, og 3.700 nye titler ble publisert. Ifølge nyere undersøkelser viser det seg at 90% av Norges befolkning leste en bok i 2002, mens 40% mener de har lest mer enn 10 bøker. 20% har kjøpt bøker over internett samme året. (*J.W. Cappelens Forlag AS 175 år i 2004* 2004)

3.3 Struktur for resten av oppgaven

I kapittel 4 tar vi for oss problemområdet. Her ser vi først på Cappelens system, deretter på ønskene fra forlaget, og til slutt sammenlikner vi disse.

Deretter ser vi på hvilke dokumentformater som egner seg å bruke hos forlaget i kapittel 5. Etterfulgt av dette ser vi i kapittel 6 på hvilke supplerende standarder som kan være nyttige.

Etter dette presenteres metoden som er brukt, samt gjennomføring og funn i kapittel 7. I kapittel 8 ser vi på forskjellige måter å strukturere et manus. Deretter presenterer vi teknologi som gir støtte for de valgte formatene i kapittel 9. Oppgaven avslutter med et forslag til en ny manusflyt for forlaget i kapittel 10.

Kapittel 4

Problemområdet - Cappelens system

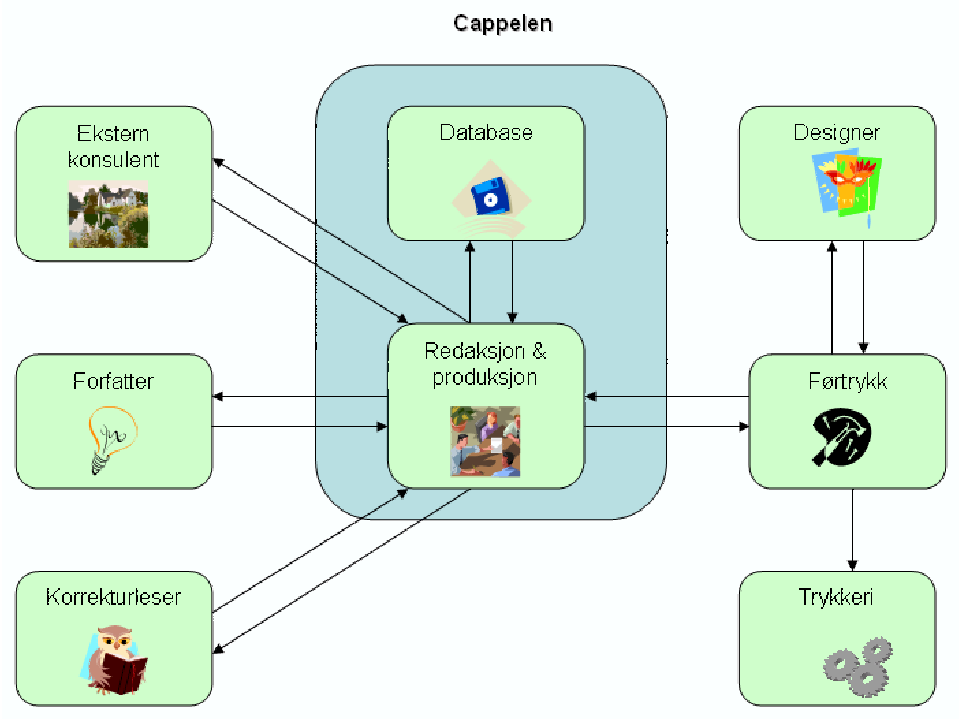
I dette kapittelet vil vi presentere det eksisterende systemet til Cappelen. Deretter tar vi for oss ønskene og kravene til det nye systemet. Til slutt sammenlikner vi disse, og prøver å finne ut hva som skal til for å forbedre systemet.

4.1 Eksisterende system

Her tar vi for oss det eksisterende systemet til Cappelen. Vi ser på flyten et manus tar gjennom forlaget, samt en kort gjennomgang av systemet Sparta.

4.1.1 Manusflyt

Vi skal se på prosessene et manus gjennomgår fra det kommer fra forfatteren til det sendes til trykk. Vi har valgt å kun ta for oss manusflyten til bøker som inngår i kategorien skjønnlitteratur i første omgang. Cappelen har en egen redaksjon for denne kategorien. Når vi har kommet fram til en bedre arbeidsflyt kan vi etter hvert få med de andre sjangrene også. Figur 4.1 på neste side er en oppsummering jeg har kommet fram til basert på to semistrukturerte intervjuer, og 3 eposter (Se 7.3 på side 57 for nærmere beskrivelse). Den er et kart over den eksisterende manusflyten. En mer detaljert oversikt over hvilke filformater som sendes mellom de forskjellige elementene presenteres på figur 4.2 på side 20. Flyten til hvert enkelt manus varierer noe.



Figur 4.1: Eksisterende manusflyt (oversikt)

Objekter som inngår i manusflyten

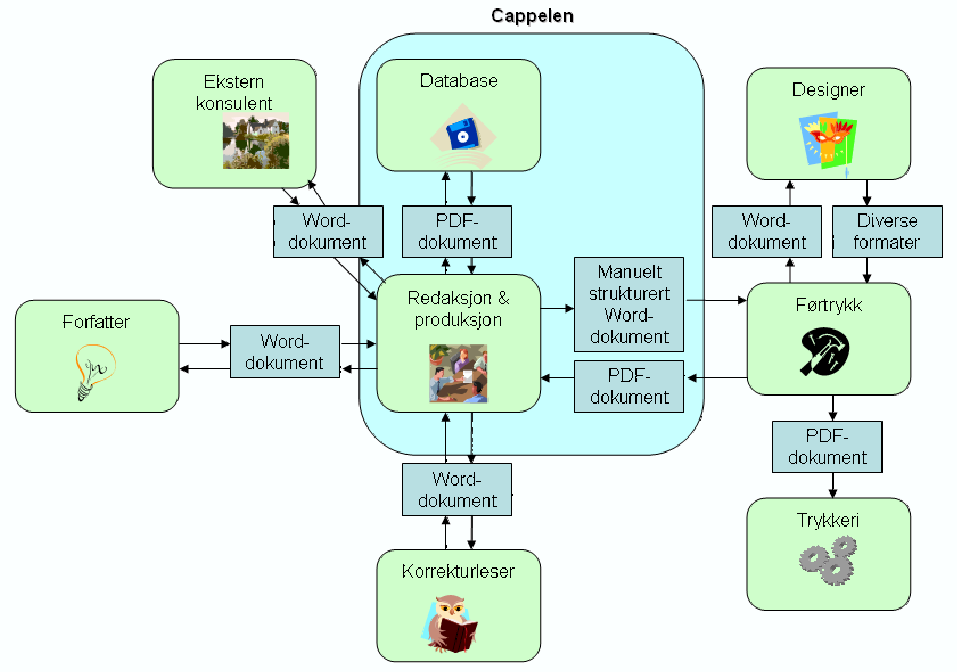
Det er 9 hovedelementer som inngår i arbeidet med et manus. En annen avdeling hos Cappelen er marked som ikke er tatt med ettersom den ikke påvirker denne delen av manusflyten

Elementer som finnes i Cappelen

- Redaksjon: Består av redaktører og en redaksjonssjef.
- Produksjon: Består av produksjonskonsulenter.
- Database: Manusene lagres i en TEAMS-database (se 4.1.2 på side 23 for mer informasjon)

Elementene som Cappelen samarbeider med

- Forfatter: Personen som har skrevet manuset.
- Korrekturleser: En person som arbeider med å finne feil i et manus.



Figur 4.2: Eksisterende manusflyt (detaljer)

- Eksterne konsulenter: Personer som leser igjennom manus, og kommer med forslag til endringer. De kan også skrive sammendrag av bøker for å lette redaktørenes arbeid. I tillegg kan de se over manus og velge å anbefale det til redaksjonen.
- Designer: En person som arbeider med den grafiske utformingen av boka, som for eksempel illustrasjoner og ombrekking. Det finnes også designere både hos Cappelen og hos førtrykket. Det varierer hvilken designer som brukes.
- Førtrykk: De som legger til stiler og brekker om boka. Det vil si at det elektroniske manuset blir delt opp i sider, og man kan få en bok ut av det.
- Trykkeri: Stedet der boka fysisk trykkes.

Manusflyten steg for steg

Her kommer en kronologisk liste over prosessene et manus skal igjennom hos forlaget.

1. Det første som skjer er at forfatteren sender sitt manuskript inn til redaksjonen i forlaget. I nesten alle tilfeller skjer dette elektronisk

over e-post. Unntaksvis får Cappelen manus innsendt på diskett eller skrevet for hånd. Håndskrevne manus må først skrives inn på maskin av eksterne firmaer som Cappelen tar kontakt med. Rundt 90 % er lagret i Word-formatet. Resten blir sendt i RTF, noen mac-varianter og andre formater. Dokumentet som blir sendt til forlaget er ikke semantisk strukturert, kun enkle formateringer er gjort.

Redaksjonssjefen fordeler manuskripter til de forskjellige redaktørene. Deretter tar redaktørene kontakt med forfatteren til manuset, og kommunikasjonen foregår deretter direkte mellom redaktøren og forfatteren. Redaktørene kan velge å sende et manus til eksterne konsulenter. De kan også sendes direkte til redaksjonen (se pkt. 3)

2. Konsulenten leser igjennom manus, og sender det tilbake til redaksjonen med kommentarer.

3. Hos redaksjonen går manuset igjennom en manusvask. Det vil si at manuskriptet nøye blir gjennomgått, og forandringer foreslås. For eksempel kan avsnitt strykes eller flyttes. Mindre endringer inngår også, som for eksempel rene språklige endringer, finpussing av setninger, syntaks og grammatikk. Dette gjøres for å øke bokens kvalitet.

4. Deretter sender redaksjonen manuskriptet tilbake til forfatteren med forslag til endringer. Dette foregår som regel på papir, men enkelte synes det er greit med elektronisk vask. Forfatteren godkjenner eller underkjenner rettelsene. Deretter innfører hun/han endringene i det elektroniske manuset.

5. Manuset blir så sendt tilbake til redaksjonen i forlaget ferdig rettet opp. Redaksjonen og forfatteren diskuterer seg fram til hvilke maler de ønsker å benytte, og hvordan boka skal se ut. Redaksjonen og førtrykk har en del maler for forskjellig utseende til bøker. Ved behov utarbeides nye maler som appelerer til et spesielt publikum.

   Før manuset sendes til førtrykk gjennomgår det også en formateringsvask. Det vil si at manuset blir bearbeidet til en standardform i Word som lett kan tilsettes forskjellige stiler. Dette er en tidkrevende, manuell prosess som gjøres av redaktørene.

6. Etter dette sender redaksjonen manuset videre til førtrykk. Der blir redaksjonens anvisninger i forhold til teksttyper fulgt. Det hender også at førtrykk kommer med egne forslag. Når dette er bestemt blir det utført en ombrekking av boka. Det vil si at den blir delt opp i sider, og antall linjer og bokstaver per side blir bestemt.

Skrifttypene og skiftstørrelser blir satt, og boka blir gjort lesbar. I illustrerte bøker der man har spalter gjøres ombrekkingen etter korrektur. Dette gjelder for eksempel grunnskolebøker som er fylt med illustrasjoner, flere kolonner og lister.

7. Alle bøkene designes, men det varierer om forlaget bruker sine egne, eksterne eller designere hos førtrykket.

8. Designer sender det ferdigdesignede manuset tilbake til førtrykk.

9. Deretter blir det ferdigombrukkede manuset sendt tilbake til forlaget. Dette dokumentet kalles 1.korrektur.

10. 1.korrektur sendes deretter til korrekturlesning på papir. Både forfatteren og en profesjonell korrekturleser utfører dette.

11. Når de er ferdige med korrekturlesingen sendes dette tilbake til forlaget, som sammenfatter og godkjenner rettelsene. Ved store uenigheter diskuterer man seg fram til en løsning.

12. Deretter sender redaksjonen rettelsene til førtrykk som retter opp det elektroniske dokumentet.

13. Dokumentet som inneholder alle endringer som nå er utført av førtrykk kalles 2.korrektur. Når det er ferdig sendes det tilbake til forlaget. Der blir 2.korrektur sjekket mot 1.korrektur. I noen redaksjoner er det på dette tidspunktet korrekturen leses igjennom i forlaget. 2.korrektur blir sjekket, lest, rettet opp og godkjent. De siste rettelsene tas gjerne på faks eller e-post med førtrykk.

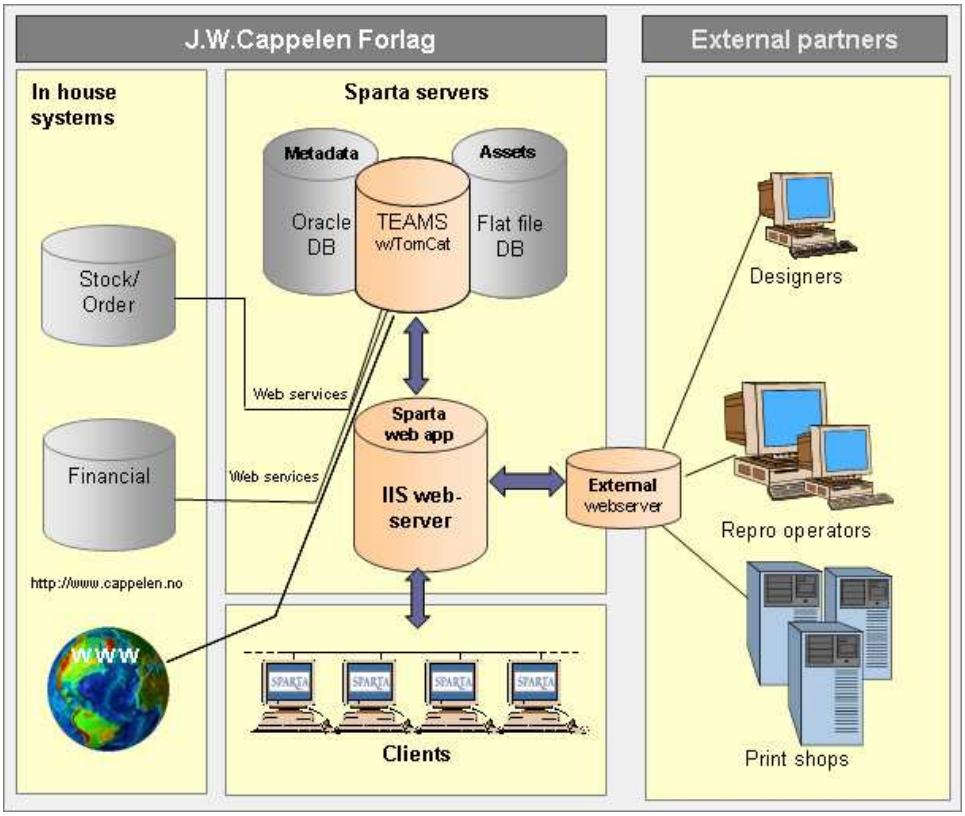
14. Når 2.korrektur er ferdig og godkjent sendes et kontrollprint til trykkeriet. Dette er for å sjekke om boka er korrekt lagret, og hvordan den vil ta seg ut.

15. Førtrykk lager til slutt en PDF av den ferdige og godkjente boka. Deretter sender de denne PDF-filen til trykkeriet og tilbake til forlaget.

16. Boka som ferdig PDF lagres i Cappelens system Sparta.

Parallelt med manusflyten blir kan forsiden til boka laget. Dette blir gjort av en designer.

De forskjellige redaktørene bruker gjerne forskjellige førtrykk som de er fornøyde med. Hvilket førtrykk de velger er avhengig av hvilke typer bøker man jobber med. Det er ofte slitsomt og ressurskrevende å bytte førtrykk, fordi man har vante samarbeidsformer. Etter å ha jobbet med et førtrykk i flere år er det ting som ikke lenger trengs å sies, noe



Figur 4.3: Sparta integrert i Cappelen

som ikke er like selvfølgelig for andre. Noen personer legger kanskje stor vekt på at boksidene skal bli gode å lese og se pene ut. I andre typer bøker, for andre målgrupper kan dette bli vurdert annerledes. Her kan det for eksempel være den pedagogiske delen som er viktigst.

Arbeidsflyten varierer noe mellom de forskjellige redaksjonene. Disse jobber individuelt.

4.1.2 SPARTA - Cappelens forlagssystem

Sparta er et nettbasert hovedverktøy for redaksjon og produksjon som Cappelen bruker i sitt arbeid. Systemet har to mål: Det første er å ta seg av publiseringsprosessen, og det andre er å kunne gjenbruke materialer i ulike medier. Programmet er bygget på toppen av Artesias Digital Asset Management System TEAMS(tm). Figur 4.3 er en figur som viser hvordan datasystemet til Cappelen henger sammen. [1]

[1]Denne figuren er hentet fra (*Sparta - A book publishing management system* 2004).

TEAMS er et system for håndtering av digital aktiva. Systemet gir mulighet for å lett finne, aksessere, dele, gjenbruke, distribuere og arkivere alle typer digitalt innhold. (*Artesia TEAMS* 2004)

Sparta forenkler hele publiseringsprosessen ved å samle all relevant informasjon på ett sted. Det holder også rede på alle hendelser ved hjelp av statusflagg, rapporter og aktivitetslister. Systemet gjør det lett å aksessere både prosjekter som planlegges, pågående prosjekter og prosjekter som er ferdige.

Sparta lagrer gjenbrukbare arbeider i en sentral database. Det er også innebygde funksjoner som gjør det mulig for eksterne samarbeidspartnere å laste opp arbeider direkte til Sparta.

Introduksjonen av Sparta i Cappelen er forlagets første steg mot en digital arbeidsflyt. For øyeblikket gir systemet bare utskrifter for tradisjonelle bokutgivelser, markedsaktiviteter og nettpresentasjoner. Fremtidige utgivelser vil kunne gi Cappelen muligheter til å lage andre produkter som for eksempel e-bøker.

Sparta er et veldig omfattende system der hver minste detalj ligger lagret. Det har et brukergrensesnitt som gir muligheter for søking, opplisting av de siste presseklipp, besøkte utgivelser, lagerinntak og klargjorte oppgaver. I tillegg har systemet en oversiktlig meny der man kan finne de funksjonene man er ute etter. Som for eksempel å lage nye prosjekter, legge inn aktiva og hente ut utgivelsesplaner.

Sparta består av fem moduler. Det første er 'metadata', her lagres egenskaper som beskriver en bok og utgivelsesprosessen. Deretter har det 'text', der tekster og salgsmateriale lagres. I 'asset' lagres ting som klassifiseres som gjenbrukbart (f.eks manuskripter, portretter og lydklipp). I 'register' lagres informasjon om utgivere og andre personer som har med prosjektet å gjøre. 'Plan' inneholder nyttige verktøy for å håndtere prosjekter, som aktivitetsplaner og rapporter.

(*Sparta - A book publishing management system* 2004)

4.1.3 Dokumentformater

Omkring 90% av dagens manus skrives i Word-formatet. Denne manusfilen blir omgjort til PDF av dagens førtrykk.

Trykkeriene tar imot manus som PDF-filer. Det finnes også et tilleggsformat som heter JDF (se avsnitt 5.4 på side 44) som en del av trykkeriene har støtte for. Dette formatet beskriver måten boka skal trykkes på.

Designerne bruker programmer som FrameMaker, Quark XPress og InDesign. Alle disse tre er ombrekning- og designverktøy som kan produsere XML. I Framemaker kan du sette på struktureringsmerkelapper. Det er blant annet laget en slik mal for DocBook (se seksjon 6.1.1 på side 47).

Oppbygning av en roman

For å avgrense oppgaven har vi valgt å ta utgangspunkt i en roman med støtte for illustrasjoner og tabeller, og ikke mer avanserte bøker som inneholder spalter og liknende. Når jeg snakker om enkle bøker videre i oppgaven mener jeg bøker som har en enkel struktur, og ikke består av flere hovedelementer enn det som blir gjennomgått her. Videre følger en skisse over hvordan en slik roman er oppbygd.

Tittel En bok har alltid en tittel, og kan ha en ekstra undertittel.

Eksempel:
"`Sundrels Legender`" (tittel)
"`Orfel fra undreskogen`". (undertittel)

Kolofon er utgiverens side med informasjon om boken. Som for eksempel kopieringrettigheter, dato, utgave, isbn, publiseringsdato og så videre. Denne siden er litt spesiell ettersom den skal oppdateres etter hvert som det trykkes flere opplag. Dette er det eneste elementet i boka som skal endres.

Eksempel:
`1.utgave`
`Copyright(c) Ruth Merethe Evang 2006`

Dedisering av en bok er vanlig. Her kan det stå hvem boka er dedikert til, og ellers annet forfatteren ønsker å si.

Eksempel:
"`Til hesten min, Falkøye`"

Innholdsfortegnelse kommer deretter. Dette er oversikten over hvor de forskjellige kapitlene befinner seg i boka. Ikke alle romaner har dette.

Eksempel:

```
1. Undreskogen ............ s.3
2. Space pizza ............ s.11
3. Zoppitappene kommer! ... s.19
4. Møtet med hovtussene ... s.24
5. Unigonens historie ..... s.31
6. Maskinen ............... s.35
7. Flukten ................ s.45
8. Tilbake til Tufisan .... s.54
9. Orfel fra Undreskogen .. s.59
10. Ritualet ............. s.66
```

Forord - I forordet kan forfatteren skrive nesten hva han/hun vil.

Eksempel:
En oversikt over hovedpersonene i boka.

Deler - En bok kan være inndelt i forskjellige deler. Hver del kan ha en tittel, og flere kapitler.

Kapitler - Hver del kan være inndelt i forskjellige kapitler. Hvert kapittel har som regel et tittel.

Eksempel:
"**`Kapittel 1 - Undreskogen`** (tittel)
`Rundt en fjern sol i en helt annen galakse enn de vi kjenner til, kretset en planet som på mange måter kunne minne om Jorden. Planeten het Tufisan og den hadde mange merkelige innbyggere. Fra verdensrommet så planeten blå ut, med grønne og røde ujevnheter på overflaten...` (avsnitt)"

Avsnitt - Hvert kapittel er inndelt i avsnitt. Det første avsnittet i hvert kapittel, eller etter tom linje, skal ikke ha innrykk etter norsk typografisk standard.

Eksempel:
"`Nede på planeten bodde en spurlap som het Spallerako. Han bodde i et passe stort hus på en firkantet øy omgitt av varm lava. Han bodde sammen med søsteren Spalleraka og foreldrene deres. Huset var laget av et stoff som kom fra de forskjellige plantene på øya.`"

Mer avansert strukturerte bøker

Andre bøker kan inneholde flere elementer. Her har jeg listet opp noen av de vanlige elementene en mer avansert strukturert bok kan ha.

Illustrasjon En bok kan inneholde forskjellige illustrasjoner

Tabell Tabeller kan ofte forekomme for å presentere data på en oversiktelig måte.

Celle En tabell kan inneholde flere celler.

4.2 Ønsket systemstøtte

J.W. Cappelens Forlag AS er i ferd med å forbedre sin nåværende arbeidstruktur, ved å ta i bruk nye datasystemer og formater som nå er tilgjengelig på markedet. Allerede har det skjedd mye i forlaget, men de er interessert i å komme enda lengre. Forlaget vil prøve å oppnå en enklere og billigere produksjon av bøker. Per i dag går mye tid med på arbeidsoppgaver som kunne vært forenklet eller unngått ved bruk av moderne metoder og verktøy.

Målet med denne oppgaven er å komme fram til en manusflyt som kan hjelpe til med å forenkle og automatisere manusarbeidet. Vi vil finne en arbeidsmåte og en arbeidsfordeling mellom forfatter og redaksjon som gjør det mulig å produsere bøker billigere og enklere, samt finne teknologi som støtter dette.

4.2.1 Krav til det nye systemet

Forlaget er opptatt av å komme fram til teknologi som gir stor støtte for to forskjellige typer gjenbruk. Den første typen gjenbruk er bøker som skal gis ut i ny utgave. Det er ofte bøker skal gis ut på nytt med små forandringer. For eksempel kan hele serier gis ut på nytt med ny forside og ny stil. I dag er det førtrykk som tar seg av dette. Cappelen sender inn de endringene de vil ha utført, og førtrykk utfører det. Hvis redaktøren kunne gjort dette selv på en effektiv måte, vil det spare forlaget for mye ressurser. Av og til skal også gamle bøker som ikke finnes elektronisk gis ut på nytt. Da er man nødt til å skanne bøkene for å få dem inn i det digitale systemet.

Den andre typen gjenbruk Cappelen ønsker å benytte seg av er krysspublisering. Det vil si at man ønsker å kunne overføre bøker til andre formater som brukes på forskjellige platformer. For eksempel kunne det være interessant å eksportere en bok til HTML for bruk på verdensveven, eller til en håndholdt PC som man kan lese i på toget. Det er også nyttig å kunne eksportere til andre kjente formater som for eksempel PDF.

Cappelen har kommet fram til at de kan redusere kostnader ved å flytte større del av arbeidet med et manus inn i forlaget. Istedetfor at førtrykk tar seg av opprettingen etter korrekturlesing og genererer PDF-filer for korrektur, vurderer de å gjøre dette innad i forlaget. Cappelen betaler mye for feil som kunne vært rettet opp tidligere. Jo senere en feil blir oppdaget, desto dyrere er det å rette den opp. Derfor vil forlaget gjøre ferdig korrekturen før manuset sendes til førtrykk. Forlaget ønsker å finne programmer og arbeidsmetoder for å brekke om et dokument for korrekturbruk. Hvis Cappelen i tillegg kunne legge til egen JDF-fil (Se seksjon 5.4 på side 44) vil de kunne spare ytterligere kostnader.

Det nye systemet skal ta utgangspunkt i idéen om at eletroniske bøker skal leve i tusenvis av år framover i tid. Derfor er det viktig å finne et system som gjør at bøkene fortsatt er tilgjengelig om mange år. Det er også et ønske om å bruke internasjonale standarder for å øke samarbeidet med blant annet trykkerier, førtrykk, andre forlag, og privatpersoner.

Det som også er viktig er å ikke knytte forlaget til et enkelt proprietært format eller til en enkelt samarbeidspartner, men å gi forlaget frihet til å endre datasystem eller arbeidsstruktur på et senere tidspunkt.

4.2.2 Innspill fra ansatte i forlaget

Vi la fram problemstillingen på et møte med noen av Cappelens ansatte, og fortalte om prosjektet. De syntes det hørtes interessant ut, og likte tanken på å få bedre økonomi, kvalitet og effektivitet. Men de hadde også en del bekymringer angående det nye systemet som vi skal ta opp her, slik at vi kan prøve å unngå dem.

Kvalitetstap

De var redde for at det skulle bli kvalitetstap på bøkene når datamaskinen var involvert i bøkenes oppbygning. Med automatisk ombrekking hadde de erfaringer med at datamaskinen kunne generere bøker som var så å si uleselige, og man kunne bli sjøsyk av å lese dem. Dette var fordi skriften var blitt dratt utover der det trengtes for å fylle linjene og sidene på en måte maskinen mente var best. De mente det var viktig å ha et menneske som kunne kvalitetssikre bøkene før de blir sendt til trykkeriet.

I en eventuell ny manusprosess er det er ikke snakk om at maskinen skal brekke om boka automatisk uten at et menneske kontrollerer resultatet. Vi kan brekke om bøkene automatisk for korrekturlesning, men ikke når boka skal sendes til trykk.

Forfatterens interaksjon

De fleste forfattere vil være med på å bestemme en del av designet til sine egne bøker. Derfor er det viktig å kunne produsere noe underveis som viser hvordan en bok kommer til å bli til slutt.

Hvis Cappelen velger å bruke et strukturert lagringsformat kan dette fakisk hjelpe til med å bedre samarbeidet med forfatteren. Ved å bruke et strukturert format kan man gjennom hele prosessen, fra manuset er ferdigstrukturert til det sendes til trykk, sette eksterne stilark til bøkene. Og man vil da kunne følge med på hvordan manuset ser ut hele tiden.

Lite fleksibilitet

En annen ting de bekymret seg over var at det skulle bli strengere regler for oppsettet av en bok. De hadde ofte problemer med datasystemet Sparta fordi alle felter måtte fylles ut. Av og til fantes det ingen informasjon til de forskjellige feltene, men likevel måtte de fylles ut.

De var bekymret for at det skulle bli slik med bøkene også hvis de måtte følge en standard. Enkelte forfattere eksperimenterer litt med de forskjellige sjangrene, og det vil forlaget ikke legge en stopper for. Redaksjonen frykter at det vil bli mindre frihet ved å innføre bruk av et semantisk språk.

Vi skal gjøre så godt vi kan for å finne fleksible løsninger som ikke skal sette en stopper for forfatterens kreativitet.

Mer arbeid

Ved å flytte arbeidsoppgaver fra trykkeriet og inn i Cappelen fryktet redaksjonen at det vil bli mer arbeid for de ansatte ettersom all korrekturoppretting skjer hos den som sitter på det designede manuset. Dette er en designer som enten er ekstern, hos Cappelen eller hos førtrykket i dag. Det må ansettes ekstra personer som er kompetente til å ta seg av dette.

Hvorfor romaner?

En av de datavante redaktørene syntes det var merkelig at vi hadde konsentrert oss om romaner i dette prosjektet. Hun mente at man ville få større nytte av dette i mer avanserte strukturerte bøker. Dette er vi enig i, men for å forenkle prosjektet mest mulig i startfasen, valgte vi å ta for oss skjønnlitteratur.

4.3 Eksisterende system sammenliknet med ønsket system

I denne seksjonen setter vi det eksisterende systemet opp mot det ønskede systemet. Vi avdekker steder i manusflyten som kan effektiviseres, og peker på løsninger for å utføre dette.

4.3.1 Gjenbruk

Først tar vi for oss gjenbruk av bøker når de skal gis ut i nye utgaver. Per i dag brukes PDF-formatet for lagring. Dette er et format som ikke er spesielt egnet for gjenbruk, ettersom det ikke er en enkel prosess å komme tilbake til et skrivbart dokument likt det man startet med. Det er tid og ressurser å spare på å ha et effektivt gjenbruk av bøker.

En måte å fremme gjenbruk på kan være å endre lagringsformat. Hvis man kan komme fram til et strukutrert lagringsformat vil man kunne endre stilark på en enkel måte. I tillegg bør det lagres i et format som man lett ekstporteres til andre formater senere.

En annen form for gjenbruk er å konvertere mellom forskjellige formater som kan brukes på ulike platformer. I dette tilfellet er det også en fordel å bruke et strukturert lagringsformat. Dette er for at programvare skal kunne forstå hvordan de forskjellige elementene skal overføres til det nye mediet.

I tillegg til det strukturerte innholdet er det nyttig å kunne lagre den grafiske utformingen til manuset adskilt fra innholdet. Ved å skille mellom innhold og grafisk utforming åpner man for enklere gjenbruk. Man kan for eksempel si at en tekst er et avsnitt. I utseendeinformasjonen kan man skrive at alle avsnitt skal være presentert på en spesiell måte. Hvis man blander grafisk utforming og innhold kan det være vanskelig å sette forskjellige stiler til dokumentet.

4.3.2 Rettinger fra korrekturlesning inn i forlaget

Per i dag rettes all korrektur hos førtrykk. For at forlaget skal kunne gjøre dette selv, er man avhengig å kunne få ut en papirkopi av boka som er behagelig å lese. Denne trenger ikke være helt perfekt slik den må være før trykking. Hvis man endrer til et strukturert lagringsformat vil det bli enklere å legge forskjellige stiler til boka. Ved å kunne legge til en bestemt stil tidlig i arbeidsfasene vil man kunne få ut et resultat som lar seg lese greit på papir, og som indikerer hvordan den ferdige boka kommer til å se ut. Man trenger å brekke om boka, men en automatisk ombrekking vil være akseptabel for korrekturlesing. For å kunne produsere det endelige resultatet er man avhengig av å ha en grafisk designer i tillegg mener IT-sjefen ved Cappelen.

Ved å gi Cappelen flere arbeidsoppgaver, trengs det ekstra personell. Det må ansettes ekstra personer til å ta seg av korrekturopprettingen.

4.3.3 Strukturering

Når en forfatter sender sitt manus inn til Cappelen er dokumentet semantisk ustrukturert. Det er kun enkle formateringer som er gjennomført. Forfattere får beskjed om å levere mest mulig ren tekst for at det skal være lettere å behandle. De får også beskjed om at formateringer de gjør kommer til å slettes.

Etter at dokumentet er ferdig korrekturlest skal en mal benyttes. For at boka skal kunne benytte denne malen må det gjøres en rekke manuelle endringer i dokumentet. Dette er en tidkrevende prosess som utføres av redaksjonen hos forlaget, og det er sannsynligvis tid å spare på å effektivisere denne prosessen. En av redaktørene har laget sine egne makroer til Word som brukes for å rense opp enkelte vanlige formateringer som forfatteren har gjort underveis.

Dette er en vanskelig del av manusflyten som vi ikke har funnet noe fasitsvar på. Selv om vi har funnet artikler som omhandler liknende situasjoner, har vi ikke sett noen som løser dette problemet. I seksjon 8 på side 63 diskuterer vi hvordan dette kan gjøres på en mer effektiv måte. Redaksjonen i forlaget har liten tro på at forfattere klarer å levere fra seg ferdig strukturerte dokumenter i henhold til en standard. Men de kan tenke seg at noen av oversetterne ville kunne ta på seg å strukturere sine arbeider. Eventuelt kan utvalgte personer i redaksjonen gjøre det.

4.3.4 Fremtidsrettet

Det er et ønske om å tenke fremtidsrettet i dette prosjektet. Vi kan tenke oss at bøkene skal være lesbare om tusen år. Det er ikke sikkert at det godseide formatet PDF fortsatt er lesbart. Standarden kan være glemt, og dokumentasjonen tapt. Det er nok større sannsynlighet for at man vil kunne lese ren tekst som er kodet med nåtidens teknologi. Det kan være mindre smart å gjøre dette i et fremtidsrettet perspektiv. Det er en fordel å bruke formater som også er åpne, frie og tekstbaserte. Hvis dokumentasjonen til formatet er offentlig tilgjengelig, kan det være lettere å arbeide med dokumenter skrevet på denne måten.

Hvis man benytter seg av et semantisk strukturert format vil det åpne for mange muligheter i fremtiden som nevnt i innledningen. For eksempel hvis man har strukturert en kokebok opp i detaljer som målenheter og ingredienser. Man vil kanskje etter hvert kunne oversette dette maskinelt til andre språk. Man vil til og med kunne oversette fra en målenhet til en annen.

4.3.5 Utfordringer

Vi har kommet fram til at Cappelen vil nå flest av sine mål ved å endre til et semantisk filformat som også gjerne er åpent, fritt og tekstbasert. Derfor vil vi konsentrere oss om dette videre i oppgaven. Vi trenger å finne ut hvilket filformat og hvilke standarder som egner seg best for Cappelens system.

Vi må komme fram til ny programvare, og en ny manusflyt som støtter det nye formatet. I tillegg må vi finne en effektiv måte å komme fra et ustrukturert dokument til det strukturerte formatet. Vi har altså disse tre utfordringene:

1. Finne et nytt, strukturert lagringsformat som egner seg for Cappelens situasjon
2. Komme fram til en manusflyt som støtter det nye formatet. Den største utfordringen er hvordan man skal komme fra et ustrukturert dokument til det nye formatet
3. Finne programvare som støtter det nye formatet

Kapittel 5

Dokumentformater

Det nåværende lagringsformatet til forlagets bøker er PDF (se seksjon 5.2.1 på side 35). I dette kapitelet undersøker vi hvilke behov forlaget har, og hvilke lagringsformater som kan dekke disse. Vi ser også på om det er PDF-filer som er det optimale formatet for arkivering og arbeidet med et manus.

5.1 Ønskelig format

I denne seksjonen ser vi på hvilke egenskaper det nye formatet bør ha.

5.1.1 Semantikk

Det første kravet som stilles til det nye formatet er at det skal ta vare på den semantiske informasjonen. Å ta vare på semantikken vil si at datamaskinen vet hvilken rolle de forskjellige delene av teksten har. I et manus skal derfor maskinen vite hva som er overskrifter, avsnitt og så videre. Motstykket til et semantisk lagringsformat er det som ikke inneholder semantikk. Et eksempel er et format som kun tar vare på den grafiske utformingen i tillegg til teksten.

Det viktigste man oppnår ved å bruke et semantisk format er en enklere form for gjenbruk. I bokbransjen hender det at bøker blir trykket opp i flere forskjellige versjoner eller utgivelser. Nye utgivelser kan inneholde oppdateringer og rettelser. Ved å ta vare på semantikken kan man lett generere nye bøker fra gamle manuskripter uten å måtte gjennomgå en større manuell prosess.

Ved konvertering til andre formater og medier er det også viktig å ta vare på semantikken. Hvis man er interessert i å gjenbruke en tekst i forskjellige medier er det viktig at semantikken forteller hvordan de forskjellige delene av teksten skal overføres til det nye mediet.

5.1.2 Grafisk utforming

Vi ønsker også enten et format som kan ta vare på den grafiske utformingen, eller å ha eksterne stilark som gjør dette. I tillegg er det ønskelig å skille mellom innhold og utseendet. Grunnen til at vi ønsker dette er at det blir enklere å arbeide med et manus. Man kan lett definere et stilark til et strukturert dokument. For eksempel kan man definere størrelse og skrifttype for de forskjellige elementene et manus inneholder. Ved å skille utseende fra innhold vil man få bedre oversikt, og lettere kunne endre grafisk utforming senere. Dette er fordi all informasjon om utseendet er lagret på ett sted, og ikke for hver enkel tekstbit i dokumentet. Hvis vi benytter oss av dette vil vi kunne generere manus med forskjellige stilark underveis i både skriveprosessen og korrekturrundene, noe som kan gjøre arbeidet med et manus mer behagelig.

5.1.3 Fritt

Det kan oppstå problematiske situasjoner ved bruk av et godseid format med mange restriksjoner. Hvis formatet er knyttet til en produsent som går konkurs, kan det få store følger. Formatet kan bli solgt til andre firmaer, og forlaget kan få et dårligere tilbud for bruk av det. Formatet kan også selges til Cappelens konkurrenter, som da setter forlaget i en dårlig situasjon. Man kan også tenke seg en situasjon der man sliter med dårlig programvare, og ikke har mulighet for utskifting ettersom alt er lagret på denne produsentens godseide format. Det kan også hende man ikke får tak i konverteringsverktøy, slik at man vil bli bundet til dette formatet for resten av manusets levetid. Og det kan bli vanskelig å gjenbruke manus i andre medier.(Pepper 2001)

Et annet problem som kan dukke opp ved bruk av godseide formater er at det stadig kommer nye oppdateringer av programmene som produserer disse formatene. Ofte må man kjøpe den nyeste versjonen av et program for å kunne lese filer som er laget med det. Dette er svært ressurskrevende, både med tanke på tid og penger. Men hvis formatet kun blir brukt innad i forlaget kan man klare seg med eldre versjoner av det.

I et fremtidsrettet perspektiv kan det derfor være lurt å bruke et så fritt format som mulig, slik at man ikke binder seg til en enkelt leverandør eller godseid format. Det finnes mange forskjellige lisenser, og det er ønske om å bruke et format som ikke har restriksjoner for bruk eller distribusjon av dokumentene på dette formatet.

5.1.4 Åpent

Et åpent format er et kjent format med offentlig tilgjengelig dokumentasjon. Det motsatte er lukkede formater, som har en hemmelig struktur, og det finnes ingen tilgjengelig dokumentasjon for dette formatet.(Hannemyr 2003)

Å gjøre bruk av et lukket format kan også vise seg å være ugunstig med tanke på fremtiden. Kanskje det ikke finnes programvare for dette formatet om 100 år. Hvis strukturen er hemmelig, og kodet uforståelig, kan det bli nærmere umulig å få tak i tekst lagret på denne måten uten å bruke den tilhørende programvaren. For å unngå at elektroniske manus går tapt bør man derfor velge et åpent format.

5.1.5 Fleksiblitet

Det er ønskelig at formatet vi velger skal kunne være utvidbart. Med dette mener vi at formatet ikke er bundet til et sett med elementer, men fleksibelt slik at man kan definere elementer eller klasser for eget bruk. Dette er viktig for å kunne definere elementer som dekker vårt behov når det gjelder lagring av innhold i bøker.

5.1.6 Tekstbasert

Jeg tror at det er størst sannsynlighet for å kunne lese ren tekst langt inn i fremtiden, i forhold til å lese dokumenter kodet på en annen måte. Derfor kan det være lurt å velge et format basert på ren tekst.

5.2 Kjente formater

I denne seksjonen ser vi på en rekke kjente og mye brukte formater, og hvilke egenskaper de har.

5.2.1 Adobe Portable Document Format (PDF)

PDF-formatet er et åpent, godseid format som kontrolleres av Adobe, men det kan fritt brukes. Det som er positivt med PDF er at det er åpent. I tillegg er PDF et format som ikke skiller mellom innhold og grafisk utforming. Det lagrer heller ikke semantisk informasjon, og egner seg derfor ikke spesielt som arkiveringsformat for bøker. I dette formatet blir både skrifttyper, bilder, grafikk og utseende på andre måter lagret, og dokumentet blir presentert likt på alle platformer (*Adobe PDF* 2004). Men dette er avhengig av at klienten enten har de samme fontene som dokumentet bruker, eller at fontene bakes inn i PDF-dokumentet. Et stort

problem med PDF er at det ikke har støtte for større gjenbruk. Det er ikke utvidbart, og er heller ikke basert på ren tekst.

Et PDF-dokument kan inneholde kombinasjoner av tekst, grafikk og bilder. Dokumentets utseende et beskrevet av en PDF-innholdsstrøm, som inneholder en sekvens med grafiske objekter som males på siden. Dette utseendet er fullt ut spesifisert. Alle avgjørelser som har med grafisk utforming og formatering har allerede blitt tatt av applikasjonen som genererer innholdsstrømmen.(*PDF Reference, fifth edition* 2005-)

5.2.2 LaTeX

LaTeX er et åpent og fritt markeringsspråk som inneholder et bibliotek med TeX-makroer og er basert på ren tekst. I praksis er LaTeX høynivåversjon av TeX som er enklere å bruke. Formatet er utvidbart ved at man kan lage sine egne biblioteker. I LaTeX kan man skrive strukturerte filer, uten å tenke alt for mye på grafisk utforming. Det er altså en blanding av et struktureringspråk og et grafisk utformingsspråk. Språket tar vare på strukturen, men likevel kan man angi grafisk utformingspesifikke elementer som fet skrift eller linjeskift. Men vi er interessert i et språk som skiller totalt mellom struktur og utseende. Det finnes både gratis og betalte versjoner av TeX. I tillegg er verktøy for LaTeX ofte kompliserte å bruke sammenliknet med Word og annet tekstbehandlingsverktøy.

StarTeX (Starter's TeX) er et program som bygger på TeX. Dette er beregnet for å skrive kortere rapporter, og er mer robust enn LaTeX. Det er struktureringsorientert, og benytter en notasjon som minner om HTML (Langmyhr 1997). Programmet inneholder kun kommandoer som man trenger for å skrive en enkel studentrapport, noe som sannsynligvis ikke vil holde for Cappelen.

5.2.3 Rich Text Format (RTF)

RTF-spesifikasjonen er en metode for å kode formatert tekst og grafikk til enkle overføringer mellom applikasjoner (*Rich Text Format (RTF) Version 1.5 Specification* 2004-). Formatet er åpent, men eies av Microsoft, og det skiller ikke mellom utseende og innhold. Det kan ta vare på noe semantisk struktur, og er tekstbasert. Det er ikke utvidbart i seg selv, men man kan definere egne maler.

5.2.4 Microsoft Word (DOC)

MS Word er en grafisk tekstbehandler med et noe avansert, men intuitivt brukergrensesnitt. Dette programmet produserer blant annet DOC-filer, HTML, XML og RTF. Formatet på disse filene er både godseid og lukket.

Programmet er mye brukt av både privatpersoner og bedrifter ettersom det har stor utbredelse, og det finnes på de fleste PC-er.

Grafisk utforming og innhold lagres sammen i Word. Man har mulighet for å lagre semantisk informasjon med programmet ved hjelp av stiler, men det er få som benytter seg av dette. Jeg tror at folk stort sett ikke er klar over at det er forskjell på en overskrift og en fet tekst i størrelse 16. Jeg antar at de fleste derfor ikke benytter seg av dette, noe som kunne forenklet konverteringen til et senere strukturert format. Men dokumentet støtter fullt ut semantisk lagring.

Ettersom de fleste forfatterne til Cappelen bruker Word, er det naturlig å ta utgangspunkt i dette formatet. Men det oppfyller ikke alle våre krav, og er derfor ikke ønskelig som sluttprodukt. Istedet må vi finne en måte å konvertere fra Word til et semantisk format. Formatet er ikke utvidbart i seg selv, men man kan definere egne maler og strukturerte elementer.

5.2.5 HyperText Markup Language (HTML)

HTML er et format som brukes for publisering av hypertekst på verdensveven. Det er både et fritt og åpent format. Det er både et grafisk utformingsspråk og et strukturspråk, men skiller ikke mellom disse. I HTML 4.0 er ca 80% av språket et strukturspråk, mens 20% er et grafisk utformingsspråk. Strukturelementene i språket er begrenset, ufleksible, og dekker ikke behovet for å lagre en fullstendig bok på en god måte. Det er heller ikke utvidbart, og egner seg derfor ikke. Men ved hjelp av eksterne stilark (for eksempel CSS) kan man definere egne klasser som kan hjelpe til med å gjøre formatet fleksibelt. Formatet er basert på ren tekst. Det har gjennomgått store forandringer i løpet av sin levetid. I lengre tid har man hatt problemer med ulike HTML-elementer for forskjellige nettlesere. XHTML er XML som følger HTML DTD-en, [1] og er nesten det samme som HTML 4.0.

5.2.6 eXtensible Markup Language (XML)

XML er både åpent, fritt, semantisk og basert på ren tekst. XML kan både brukes som et markeringsspråk, et metamarkeringsspråk, og som et språk i seg selv. Det er basert på et subset av språket SGML (the Standard Generalized Markup Language, ISO 8879). Elementer er både lagt til og fjernet. For eksempel hyperlenker finnes i XML men ikke i SGML. Med XML kan man designe sitt egenlagde språk for ulike typer dokumenter. Kort sagt gjør man dette ved å definere hvilke attributter og entiteter som er lovlige, og på hvilken måte disse kan brukes.

[1] Mer om XML og DTD i seksjon 5.3 på neste side

	Fritt	Åpent	Semantisk	Skiller layout og innhold	Utvidbart	Tekst-basert
PDF	Nei	Ja	Nei	Nei	Nei	Nei
LaTeX	Ja	Ja	Noe	Noe	Noe	Ja
RTF	Nei	Ja	Noe	Nei	Noe	Ja
DOC	Nei	Nei	Noe	Nei	Noe	Nei
HTML + stilark	Ja	Ja	Noe	Noe	Noe	Ja
XML + Stilark	Ja	Ja	Ja	Ja	Ja	Ja

Figur 5.1: Oversikt over de forskjellige formatenes egenskaper

Den største fordelen med XML er den sterke strukturingen. Alle elementer må struktureres, og man kan angi egne struktureringslover som koden må være skrevet i henhold til. I tillegg er grafisk utforming og struktur strengt adskilt. For å vise fram et XML-dokument er det nødvendig med eksterne stilark som beskriver utseendet. Strukturen er utvidbar slik at det kan benyttes for å lagre avanserte bøker.

5.2.7 Konklusjon

Vi ser av figur 5.1 at XML er det eneste formatet som oppfyller alle våre krav, og passer derfor best for forlaget. For å dekke den grafiske uformingsdelen i XML trenger man eksterne stilark.

5.3 Extensible Markup Language (XML)

XML er et språk for strukturering av informasjon. Det kan både brukes som et markeringsspråk (eng. markup language), et metamarkeringsspråk, og et språk i seg selv. Et metamarkeringsspråk er et språk som kan brukes for å definere andre markeringsspråk.(Flynn 2003) XML er

basert på et subset eller et utdrag fra SGML, med tillegg som for eksempel hyperlenker. Ved hjelp av XML kan man bli i stand til å introdusere den semantiske verdensveven.(*XML* 2004)

5.3.1 XML blir til

HTML har fungert tilfredsstillende på verdensveven i 10-15 år. Den har oppfylt de tre viktigste kravene som stilles til et nettverktøy. Det har støtte for lenking, er enkelt å lage og å lære seg, og sist men ikke minst er det platformuavhengig. Men etter hvert som verdensveven utviklet seg dekket ikke HTML alle behov lenger.

Etter hvert ble man interessert i å utvikle et språk med bedre muligheter for tilpasning og vedlikehold, og som tok vare på mer semantisk informasjon. HTML er et lite fleksibelt språk som ikke er godt tilpasset i alle situasjoner. Det ble derfor ønskelig med et nytt språk som bedre kunne tilpasses, og XML ble utviklet.

XML fortsatte der HTML ikke strakk til, ved at det ble et utvidbart og fleksibelt språk som tar vare på semantikken. XML har fortsatt støtte for lenking, er enkelt og platformuavhengig. Den største styrken med XML er at språket er sterkt semantisk strukturert. Ved å definere et element som heter `<pris>` kan man si at et element faktisk er en pris, og ikke bare er et tall. Hvis man i tillegg har med elementet <valuta> kan man si hvilken valuta prisen er angitt i. Dette åpner for omregning til andre pengeenheter. XML er også tilpasningsdyktig. Man kan definere sine egne elementer som gjør at folk kan lage et sett med elementer som er tilpasset sitt bruk. Sist, men ikke minst er XML godt vedlikeholdbart. Dette er fordi XML-dokumenter kun inneholder data og struktur. Utseende beskrives i separate stilark, og lenker er separate, og ikke nedgravd i XML-koden. Det er enkelt å aksessere og forandre XML.(Rees 2004-)

Et utvalg av syntaksen ble overført fra HTML til XML slik at de som brukte dette nye språket og hadde kjennskap til HTML ville kjenne seg igjen. Et eksempel er at elementer må omsluttes med krokodillegap, begynne med < og avslutte med >. Til slutt ble det bestemt at XML ikke trenger en tilgjengelig DTD[2]. (*XML* 2004)

XML ble utviklet av “the SGML Editorial Board”, og formet under World Wide Web Consortium (W3C) sine rettningslinjer. Prosjektet ble startet i 1996. I 1998 ble XML godkjent og anbefalt av W3C. Det er den høyeste status et dokument kan oppnå fra W3C. XML er stabilt, klart for å tas i bruk, og klar for å bli spredd videre.(*XML* 2004)(Treese 1998)

[2]Se avsnitt 1.4.6

5.3.2 Bruk av XML

XML består av et sett med ukompliserte regler, og gir oss en måte å beskrive og utveksle strukturerte data. Med XML beskriver man struktur og semantikk, men ikke formatteringen. Dette tar vi hånd om i eksterne stilark. Et XML-dokument inneholder ikke grafisk utformingsinformasjon, og trenger altså et stilark som tar seg av fremvisningen. Nettlesere har innebygde stilark som benyttes, og man trenger kun å definere enge stilark hvis man ønsker å overskrive denne. XML er case-sensitivt, dvs at små og store bokstaver ikke betyr det samme. <p> er med andre ord ikke det samme som <P>.

5.3.3 Gyldighet

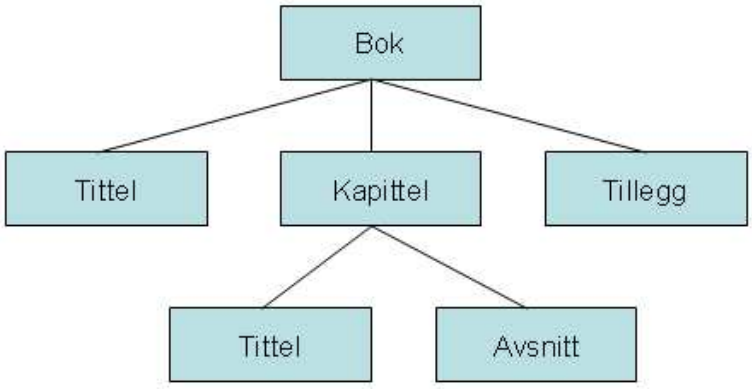
Det finnes to forskjellige grader av gyldighet. Man har dokumenter som er gyldige, og man har de som er velformet. For at et dokument skal være velformet må det følge XML-syntaks. For at det skal være gyldig må det i tillegg følge en bestemt gramatikk, som beskrives i en såkalt DTD eller XSD. 5.3.6 på side 43 (Walsh 2004)

5.3.4 Designmål

- XML skal være mulig å bruke rett fram over hele internett
- XML skal ha støtte for et bredt spekter med applikasjoner
- XML skal være kompatibelt med SGML
- Det skal være lett å skrive programmer som prosesserer XML-dokumenter.
- Valgfrie funksjoner i XML bør være så få som mulig, ideelt null.
- XML-dokumenter skal være menneskelig lesbare og logiske.
- XML-design skal være formelt og konsist.
- XML-dokumenter skal være enkle å lage.
- Kompakthet i XML-kode er minimalt viktig.(Bray 2004)

5.3.5 Oppbygning

Byggeklossene til XML er forskjellige typer noder. Med unntak av rotelementet kan alle nodene settes sammen flere ganger og på forskjellige måter. Disse byggeklossene er elementer, rotelementet, attributter, navneområder, prosesseringsinstruksjoner, kommentarer og tekst.



Figur 5.2: Eksempel på et XML-tre

Elementer

Den største delen av XML er elementer. Et element inneholder en starttagg, et innhold og en avslutningstagg. En tagg må omsluttes med krokodillegap, begynne med < og avslutte med >. Elementet har et navn, og kan ha attributter og innhold. Innholdet kan være både ren tekst og andre elementer. Tomme elementer trenger ikke et avslutningselement. For å si at elementet er tomt legges det til en skråstrek før avsluttende >. Et tomt element kan være `<element></element>` som kan skrives `<element/>`. Et ikke-tomt element kan for eksempel være `<element>Innhold</element>`.

Ethvert ikketomt element må ha både en starttagg og en avslutningstagg. Omsluttende elementer er ikke tillatt i SGML, og dette ble overført til XML. For eksempel er det lov til å skrive `<element1> <element2> </element2> </element1>`, men derimot `<element1> <element2> </element1> </element2>` er ikke tillatt.

Rotelementet

Det et krav at et XML-dokument må ha et unikt rot-element som ligger helt på toppen. Man kan tegne opp strukturen i et XML-dokument som et slektstre. Som regel tegner vi treet opp-ned, dvs at roten tegnes øverst. Det øverste elementet kalles rotnoden. Andre elementer som ligger under dette kan man tenke på som grener som brer seg utover fra treet. Disse kalles barneelementer av rotelementet fordi de er på et høyere nivå i treet. Figur 5.2 er et eksempel på et slikt tre.

Attributter

Attributter hører til et bestemt element, og kan gis en verdi. De gir ofte informasjon som ikke er en del av dataene. For eksempel `<fil type=“gif”>bilde.gif</fil>`. I dette eksempelet sier attributtet hvilken filtype et blide er slik at syntaksanalysereren (eng. parser) vet hva slags programvare som skal manipulere elementet. Elementer kan ha forskjellige atributter. Disse har sin plass inne i element-taggene. Et element kan ikke ha det samme attributtet flere ganger. Verdier til atributter kan enten stå i “doble” eller ’enkle’ fnutter. Dette kan være nyttig hvis informasjon inneholder fnutter (*XML Attributes* 2004). For eksempel `<artist navn=’Sigurd “Satyr” Wongraven’/>`

Navneområder

Ved definering av navn til elementer kan det oppstå navnekonflikter. Ved at to er elementer gyldige, men ulike, og har samme navn. For å unngå slike konflikter bruker man navneområder (eng. namespaces). Det plasseres først i et element, og skrives `xmlns:namespace-prefix=“namespace”` Det kan også defineres ved å bruke en internettadresse. Navnet blir ikke brukt av syntaksanalysereren for å hente informasjon. Det eneste målet er å gi navneområdet et unikt navn. Ofte bruker firmaer navneområder som pekere til nettsider som inneholder informasjon om navneområdet. (*XML 1.0 and Namespaces* 2001)

Prosesseringsinstruksjoner

I XML kan man gi instruksjoner som skal prosesseres. Dette gis til applikasjonen som prosesserer XML-dokumentet. Dette kan for eksempel være informasjon som hvordan dokumentet skal behandles, og hvordan det skal vises fram. Hver slik instruksjon er barn av et element. Instruksjonen består av to deler; målet eller navnet på instruksjonen, og data eller informasjon (Gudgin 2001). For eksempel `<?display table-view?>`

Kommentarer

Som i andre programmeringspråk er det muligheter for å skrive kommentarer i XML. Disse vil ikke påvirke koden. Dette skrives på samme måte som i HTML; `<!-- Dette er en kommentar -->`.

Tekst

Ren tekst forekommer mellom elementtagger og som attributtverdier.

5.3.6 Document Type Definition (DTD)

En DTD er et regelsett som definerer bruk av og strukturen til et XML-dokument. Her defineres hvilke elementer som kan være inneholdt i andre elementer, og hvilke attributter de forskjellige elementene kan eller må inneholde. HTML tar også i bruk en DTD. Denne ligger lagret i de forskjellige nettleserene som finnes. Derfor kan et HTML-dokument opptre forskjellig i ulike omgivelser. Brukere har ikke mulighet for å endre HTML DTD-en, men ved hjelp av XML kan folk definere elementer til sitt eget behov.(Garshol 1999)

Det finnes to typer XML-dokumenter. Man kan enten definere det som frittstående, dvs at XML-filen ikke har en tilhørende DTD. I dette tilfellet er det nettleseren som må tilby denne. Den andre typen XML-dokumenter er de som har en tilhørende DTD som ikke er innebygd i nettleserene. Denne må være tilgjengelig for at dokumentet skal kunne vises. Et XML-dokument må være gyldig i henhold til den angitte DTDen. Hvis ikke kan det ikke vises, og er et ugyldig dokument (Garshol 1999).

5.3.7 XML Schema (XSD)

En XSD er også et regelsett som beskriver strukturen i XML-dokumenter. Men denne er bygget opp på en litt annen måte. Og man har muligheter for å beskrive flere regler i denne. Min erfaring er at DTD-en kanskje er mer intuitiv, og lettere å få et overblikk over når man skal sette seg inn i det. Men XSD-en er lettere å arbeide med når man har gjort seg kjent med den. I tillegg finnes det forskjellige verktøy man kan bruke for at arbeidet med en DTD eller en XSD skal gå lettere. For eksempel har XML-SPY (se seksjon 9.2.2 på side 71) mulighet for å fremvise en grafisk representasjon av disse. En XSD er skrevet i XML.

5.3.8 Extensible Stylesheet Language Transformation(XSLT)

XSLT brukes for å overføre XML til et annet dokument. Dette blir brukt for å presentere data på en oversiktelig måte. XSLT er et transformassjonspråk, dvs et språk som transformerer data fra en representasjon til en annen. XSLT-prosessoren har muligheten for å transformere et XML-element til et annet. Den kan også transformere direkte til HTML eller ren tekst.

Et problem med XSLT kan være at det er ganske nytt på markedet. Det er fortsatt under utvikling, og vil med høy sannsynlighet gjennomgå forandringer i løpet av de nærmeste årene. Hvis dette skjer får vi håpe at de neste versjonene vil være bakoverkompatible med den versjonen Cappelen eventuelt skal benytte. Men ettersom det lagres i ren tekst vil det alltid være mulig å få ut informasjonen på en eller annen måte.

Og man vil derfor kunne lage konverteringsverktøy til det eventuelt nye formatet.

Selv om XSLT er et kraftig transformasjonspråk har det visse negative egenskaper. Det viser seg at språket er veldig kompleks. Selv for å transformere et enkelt XML-dokument må brukeren skrive et helt program. For å skrive dette må man ha gode programmeringskunnskaper. Det foregår stadig forskning etter nye og enklere overføringspråk, men de er fortsatt komplekse.

5.3.9 Cascading Style Sheets (CSS)

Et stilark det går an å bruke til et manus er CSS. Det brukes i dag for fremvisning av nettsider. Språket er kun et språk for grafisk utforming, slik at den grafiske utformingen skilles fra innholdet. Uheldigvis er det også mulig å blande CSS direkte inn i strukturerte formater, noe som gjør at språket kan misbrukes. CSS er veldig intuitivt, og kan oppfattes som lettere både å bruke og å lese enn XSL:FO (se 5.3.10). Man har mange muligheter når det gjelder CSS. Den nyeste versjonen, CSS2, har rundt 110 forskjellige egenskaper. Men denne støttes ikke enda fullt ut av noen nettlesere.

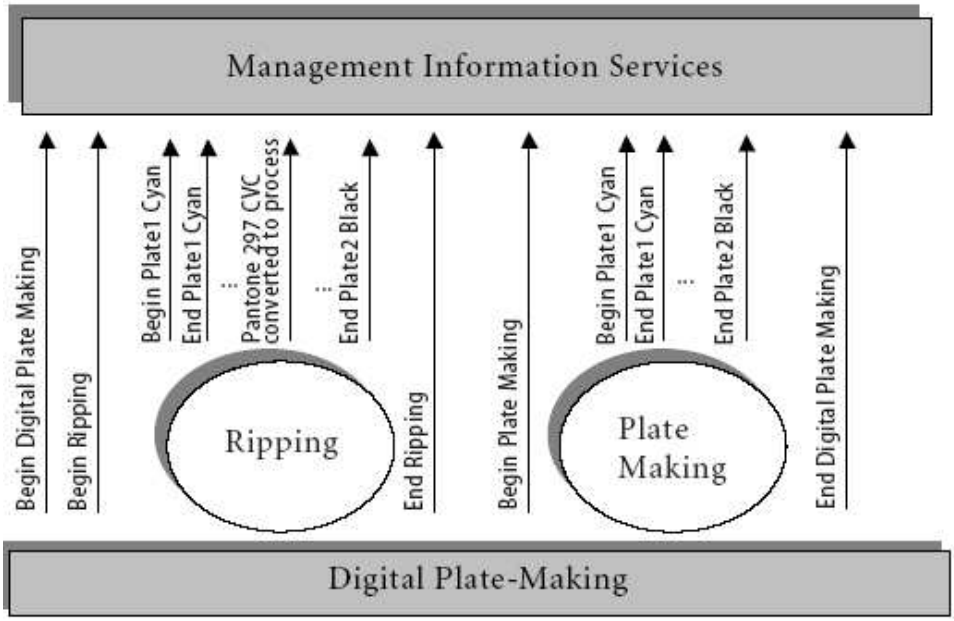
Men man kan ikke gjøre alt med CSS som man kan med XSLT. For eksempel er det bare XSLT som kan generere en innholdsfortegnelse. Noe som kunne vært aktuelt for forlaget var å bruke både XML, XSLT og CSS for å produsere et manus. Da dekker man alle funksjonene til XSLT, og man har i tillegg med CSS sin enkelhet og oversiktelighet.

5.3.10 XSL Formatting Objects (XSL:FO)

XSL:FO er en annen måte å fremvise et XML-dokument på. Det er litt av den samme idéen som CSS har, men den er utført på en litt annen måte. XSL:FO kan være litt komplisert, og koden blir ofte lang og ikke så intuitiv å forstå. CSS er et enklere språk som er lett å forstå, og koden blir liten og enkel.

5.4 Job Definition Format (JDF)

JDF er et XML-basert format som ble utviklet i et samarbeid mellom Adobe, Agfa, HEIDELBEG og Man Roland. JDF er designet for å sende informasjon mellom forskjellige applikasjoner og systemer. Poenget var å få hele trykkerier til å arbeide med individuelle arbeidsflytløsninger. JDF er en åpen XML-basert standard, og inneholder informasjon om hele prosessen med å lage blant annet en trykkeklar bok. Helt fra den kreative prosessen, til trykkeprosessen og leveringsprosessen. På denne



Figur 5.3: Kart over JDF

måten kan man bygge en bro mellom kundens syn på produktet, og produksjonens syn på fremstillingsprosessen. I tillegg kan man også definere og følge en brukerdefinert arbeidsflyt. Dette er altså en standard som brukes i arbeidsprosessene, og ikke i den ferdige boka.

En del trykkerier er interessert i å begynne å bruke denne standarden, og flere har allerede støtte for det i systemet sitt. Ved å legge ved en JDF-fil med et manus kan dette være med på å forenkle arbeidet trykkeriet har med et manus fra forlaget. Denne filen inneholder blant annet opplag (antall kopier av boka) for en bok, og hvilket format den skal trykkes i. Ved å sende dette i en JDF-fil som kan brukes direkte av trykkeriet, minsker man muligheter for menneskelige feil, og man vil kunne få en sikrere produksjon. Forlaget har tidligere opplevd å få blant annet feil antall kopier av en bok i feil format. (*Dataspråk* 2004)

Figur 5.3[3] er et eksempel på informasjon en JDF kan inneholde. Her ser man hvordan informasjonshåndtereren kommuniserer med den digitale platelagingen. Dette eksempelet inneholder detaljert informasjon om hvordan platene som skal trykkes blir laget. Dette er bare en liten del av hva en JDF kan inneholde.

[3]Hentet fra (*JDF White Paper* 2005-)

5.5 Vurdering

Vi velger å bruke XML for lagring av innhold i bøker. Dette er fordi formatet både er fritt, åpent, semantisk, utvidbart, tekstbasert og det skiller mellom grafisk utforming og innhold. Vi vil bruke XSLT som transformator fordi dette er et kraftig verktøy som man for eksempel kan bruke til å generere innholdsfortegnelse. Vi velger å bruke dette fordi vi ikke kjenner til annet verktøy med de samme egenskapene. I tillegg trenger vi eksterne stilark for lagring av grafisk utforming. Her velger vi CSS fordi det er et enkelt, lettforståelig og utvidbart språk. Hvilke stilark som benyttes er også opp til designerne, så lenge det ikke kommer i konflikt med Cappelens ønsker om å lagre data strukturert, og skille mellom innhold og formatering.

Vi ønsker også å benytte en DTD eller XSD for å validere manusene på XML-form. I kapittel 6 på neste side diskuterer vi oss fram til hvilket regelsett som passer best for Cappelen.

Kapittel 6

Standarder

Vi ønsker å benytte en åpen, internasjonal standard for boklagringen. Cappelen kunne laget sin egen DTD/XSD, men de ønsker heller å ha et tettere samarbeid med blant annet andre forlag, trykkerier og designere.

6.1 Forskjellige DTD-er og XSD-er

I denne seksjonen ser vi på en rekke forskjellige regelsett, og prøver å finne ut hvilken som passer best for forlaget.

6.1.1 DocBook

Docbook er en XML DTD som beskriver et språk som er egnet for boklagring. DTD-en var opprinnelig laget for databøker, og er derfor veldig dataorientert. Denne DTD-en er forholdsvis stor og omfatter ganske mye mer enn hva Cappelen trenger for å beskrive sine bøker. Men et subset av den er aktuell. DTD-en finnes både for XML og SGML, og vi velger å bruke XML-versjonen.

Når dette skrives er den nyeste versjonen 4.2. Docbook har vært under utvikling i mange år. Forløperen ble født i 1991, og denne har med tiden utviklet seg til den DTD-en vi i dag kaller Docbook. Den ble utarbeidet til en DTD av O'Reilly and Associates og HaL Computer Systems for å få UNIX-dokumentasjonen til et bedre format som lett kunne gjenbrukes. I 1998 tok OASIS (Organization for the Advancement of Structured Information Standards) hånd om Docbook. I Docbook kan man ikke bruke udefinerte tagger. Dette er fordi DTD-en skal være konverterbar til andre formater. Hvis man bryter med dette, mister man en vesentlig fordel med Docbook.(Brockmeier 2001)

OASIS arbeider med spesifikasjoner for industrien, som er basert på offentlige standarder som XML og SGML. De har en egen komité for utviklingen av Docbook.

6.1.2 Text Encoding Initiative (TEI)

TEI DTD-en er en forskerorientert DTD som er mye brukt i den akademiske verden for lagring av tekst og dokumenter. Den er ganske tung å sette seg inn i, men er grei å arbeide med. Denne DTD-en ble designet for å markere et hvilket som helst dokument i et hvilket som helst språk. Den har derfor en løs struktur, og bruker generisk oppdeling for å separere seksjoner fra dokumenter.

Denne DTD-en ble laget for en god stund siden, da lagringsressursene var på et mye tidligere stadium enn i dag, og det har ført til at navn på elementene er litt merkelige, og ikke helt intuitive. Men den originale versjonen av TEI hadde muligheter for å endre navn på elementer.

Fullversjonen av TEI DTD er enorm, og det ble derfor laget et subset av denne som blir kalt teixlite.dtd. I denne versjonen er ikke endring av elementnavn tillatt. (*An Introduction to the Text Encoding Initiative (TEI), DTD* 2001)

6.1.3 Office 2003 XML

Microsoft har utviklet XML-skjemaer for bruk i Office-relaterte produkter. Skjemaene inkluderer blant annet skjema for Microsoft Office Excel 2003, Microsoft Office InfoPath 2003 og Microsoft Office Word 2003. (*Office 2003 XML Reference Schemas Overview* 2005) Disse skjemaene er gratis, og kan brukes fritt. Men de er godseide, og man har ikke tillatelse til å modifiseres skjemaene.(*Legal Notice* 2005) DOC-filer kan konverteres direkte til dette formatet.

6.1.4 OpenOffice XML

OpenOffice XML er en fri standard som også er utviklet for kontorverktøy. De ble opprinnelig laget av OpenOffice, men nå er standarden under videre utarbeidelse av OASIS.(*Open Document Format for Office Applications (OpenDocument) 1.0, commitee draft 2* 2004)

Office 2003 XML og OpenOffice XML er utviklet for Office-applikasjoner, og er ikke skreddersydd for boklagring. Derfor stiller vi oss litt kritisk til bruk av disse to standardene i vårt prosjekt. Vi vil gjerne ha en standard som tar vare på all informasjon man trenger å lagre når man skal digitalisere en bok.

6.1.5 Andre DTD-er

Det finnes i tillegg en rekke DTD-er som er utviklet for biblioteker. Men disse har det fellestrekket at de kun inneholder metainformasjon om bøker, og ikke innholdet. Derfor kan ikke forlaget kun bruke en av disse

for sine bøker. Men de kan bli brukt for å lagre metadata, hvis man finner en annen standard man kan bruke til boklagring som ikke inneholder dette.

Gutenberg(*Book DTD's I* 2001) har utviklet en del DTD-er. En enklere DTD er `poemsfrag.dtd`. Denne er opprinnelig laget for å markere diktene til Robert Burns. Derfor er denne veldig simpel, og passer best til små tekster som dikt. Denne ble etter hvert utvidet til `gutbook.dtd`. Opprinnelig ble den utvidet for å ta vare på Charles Darwins bøker. En DTD som bygger på denne igjen er `gutplay.dtd` som er laget for skuespill som er inneholdt i bøker.

6.1.6 Konklusjon

Det ser ut til at DocBook er den av standardene som har det beste utgangspunktet for lagring av bøker hos Cappelen. Denne standarden er spesiallaget for å ta vare på semantisk informasjon om bøker. Den er mer intuitiv, og enklere å forstå enn TEI DTD-en. I tillegg har den støtte for mange viktige elementer som inngår i en bok. Office 2003 XML og OpenOffice XML er mer generelle DTD-er som ikke er skreddersydd for lagring av bøker, men for office-verktøy. Et pluss er at DOC-formatet kan konverteres direkte til Office 2003 XML. Men siden formatet i tillegg er godseid ønsker vi ikke å benytte dette.

6.2 DocBook

DocBook er standarden som vi vil bruke hos forlaget. Det kan diskuteres om dette er den beste standarden som finnes, men den tilfredstiller våre krav, og passet best av de vi fant. For at DocBook-standarden skulle være enklere å arbeide med brukte vi en XSD som er laget utfra DTD-en. Ved å se nøyere på den finner vi ut at den en svakhet.

6.2.1 Docbook hos Cappelen

I standarden lagres både informasjon om boka og om innholdet i samme fil. Vi ønsker å skille disse mer i dette prosjektet. I en relasjonsdatabase trenger man kun én instans av hver forfatter som kan kobles mot de forskjellige bøkene. Hvis all metadata skulle være innholdt i manuset på XML-form som følger DocBook XSD-en ville man fått mye dobbeltlagring. Her er det felter for å kunne lagre blant annet både navn, adresse og telefonnummer til en forfatter. Dette ville lagres i alle bøkene, noe som tar opp unødvendig plass og gjør at ikke alt er samlet på ett sted. Hvis forfatteren plutselig skiftet telefonnummer, måtte man endre dette i alle bøkene som forfatteren har skrevet. Derfor er det enklere for Cappelen

å lagre metadata separat. En mulighet for Cappelen er å ikke ta med metadata i selve XML-dokumentet, men heller lagre dette eksternt i Spartas metadatabase.

En annen svakhet som DocBook har sett fra Cappelens bruksområder, er at den er alt for omfattende. Den inneholder mange små detaler som kun er aktuelle for databøker.

For å løse dette kan Cappelen benytte et gyldig subset av DocBook. Det er mulig å konsentrere seg om enkelte elementer, og fjerne ikke-nødvendige elementer fra XSD-en. På denne måten vil man få en enklere XSD som er lettere å holde oversikten over.

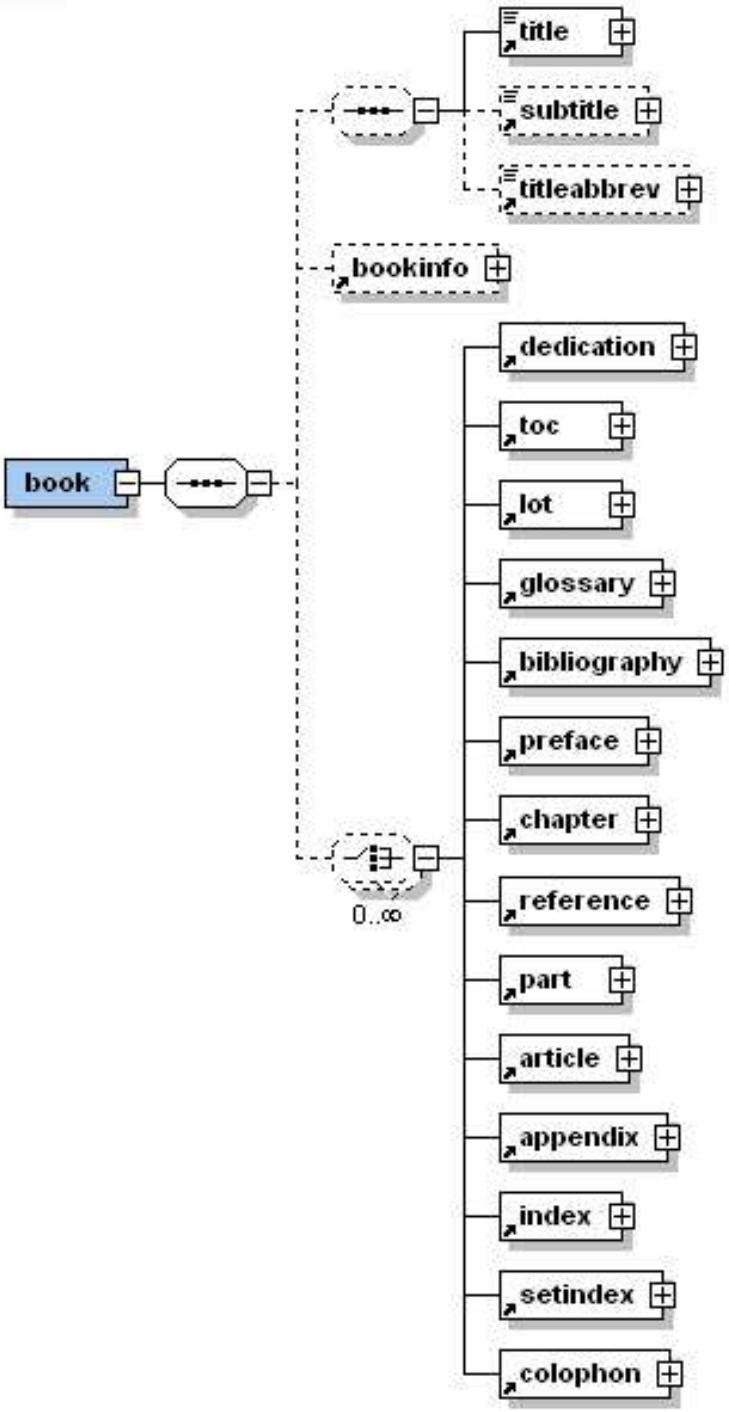
6.2.2 Tilpassing

En mulighet for Cappelen kan være å tilpasse DocBook XSD-en for forlagets behov. Hvis dette gjøres, bør det gjøres i tråd med XSD-ens regler. Det vil si at strukturen må være den samme, men valgfrie elementer kan tas bort. Hvis man endrer på strukturen vil ikke et dokument lagret på denne måten lenger være et gyldig DocBook-dokument. Vi har da brutt med standarden.

Hvis man likevel ønsket å endre strukturen bør man ihvertfall sørge for at denne er konvertérbar til DocBook. På denne måten kunne man hentet inn gyldig DocBook, konvertert dokumentet til Cappelens spesifikke XSD, og deretter eksportert dokumentet til DocBook igjen. Hvis dette er en aktuell løsning må man vurdere om omgjøringen av DocBook er så mye verd at man er villige til å holde på med denne konverteringen. Et alternativ er at Cappelen samarbeider med OASIS for å utvide DocBook dersom det skulle vise seg nødvendig. Et annet spørsmål er om det bare er arbeidet som skal gjøres i forlagets egene XSD, eller om bøkene skal lagres slik i databasen. Hvis man innfører dette kan man også legge til elementer i Cappelens egen XSD som kun skal brukes internt hos forlaget. For slike endringer vil ikke kunne transformeres til DocBook.

6.2.3 Forenkling

Det er ønskelig med en forenkling av DocBook. Det finnes en versjon av den, som heter DocBook Lite. Men denne inneholder kun artikkel-elementer, og ikke bok-elementer. Grunnen til at vi ønsker å forenkle XSD-en er at den skal bli lettere å arbeide med. Ved å forenkle denne blir det også lettere å lage en applikasjon som man skal kunne produsere bøker i også. Brukerne bør ikke bli for overbelastet med knapper eller menyelementer. Dessuten er det tungt å arbeide med denne standarden uten hjelpemidler, ettersom den er så omfattende som den er.



Figur 6.1: Strukturen i DocBook

Hvis man skal bruke et program for å skrive bøker i på DocBook-form kunne man brukt hele standarden, men kun implementert en liten del av den i programmet. Det er en omfattende oppgave å finne ut hvilke elementer man trenger å implementere, og hvilke som er overflødige.

6.2.4 Oppgradering

Hva skjer når DocBook kommer ut i en ny versjon som blir den nye standarden? Cappelen står da ovenfor tre forskjellige valg.

- Den første og enkleste løsningen er å ignorere at standarden er oppgradert. Dette fører ikke til noen endringer hos forlaget, men det bryter med samarbeidet med resten av verden.
- Den andre løsningen er å la alle bøker som er lagret i den forrige versjonen være som de er, men å lage alle nye bøker i den nye versjonen. Dette ser ut til å være en grei løsning, men det gjør at man får dokumenter i flere forskjellige formater, og at man må ha alle de brukte XSD-ene tilgjengelig.
- Det siste alternativet forlaget har er å konvertere alt det gamle til å følge den nye DTD-en, samt lagre alle nye bøker i denne versjonen. Denne omgjøringen kan kreve tid og ressurser. Men dette gjør at alt i databasen er i samme versjon, og at man beholder samarbeidet med andre parter som også bruker DocBook.

Hvis man velger å endre eksisterende dokumenter for at de skal følge den nye standarden er det høyst sannsynlig at det må endres i stilarkene som hører til disse. Derfor vil vi anbefale å kun lagre nye bøker i den nyeste versjonen.

Hva gjør Cappelen hvis standarden endres når forlaget har laget sin egen versjon av DocBook? Denne må da gjøres om slik at den er gyldig i forhold til den nye DTD-en. Det man kan gjøre er å gå igjennom den nye versjonen manuelt, og gjøre de endringer som man ønsker i forhold til denne, og i forhold til hva man har gjort før. Det man ellers kan gjøre er å lage et program som går igjennom den nye versjonen og endrer denne automatisk. Hvis man kun har fjernet elementer, og ikke endret på struktur, vil det være en grei sak å lage et slikt program. Dette programmet går igjennom og sletter elementer som ble fjernet i den forrige versjonen, og disse elementenes underelementer. Slik at man igjen får et gyldig subset av DocBook.

6.2.5 Forslag til endringsprogram

Her presenteres et forslag til hvordan et program kan gå igjennom og fjerne uønskede elementer fra DocBook XSD-en.

Vi oppretter en fil som inneholder navn på alle elementene som skal fjernes. Disse er separert med linjeskift. Deretter lager vi et program som består av fire klasser.

1. En **fil-leser** leser inn hele DocBook XSD-en, og sender den til trebyggeren. Deretter går den igjennom filen med elementene som skal fjernes, og leverer linje for linje til stripperen.

2. **Trebyggeren** bygger et globalt tre av den ferdige XSD-en, og returnerer det til fil-leseren.

3. Hver gang **stripperen** får inn et element som skal fjernes, går den igjennom treet (bredde først-søk) og sletter dette elementet, samt dets barnenoder.

4. En **fil-skriver** gjør om det ferdigstrippede treet til XSD, og skriver dette til fil.

6.2.6 Forslag til endringer av DocBook

Ettersom DocBook-standarden har visse svakheter velger vi å presentere to forslag til endringer av standarden. Disse endringene kan gjøre den mer attraktiv for bokmarkedet.

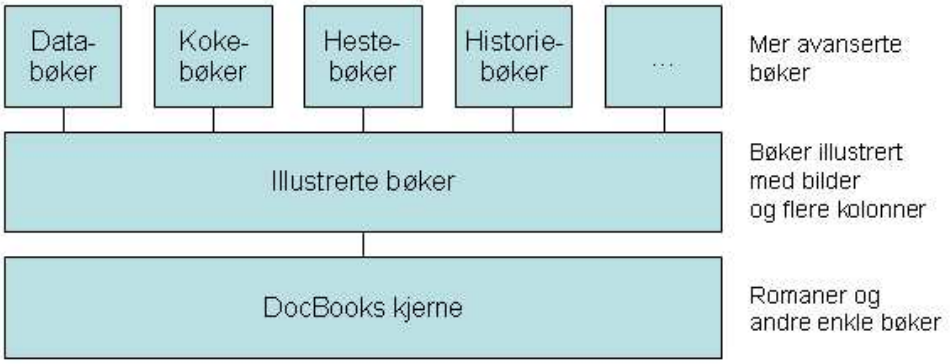
En kjerne med ekstra moduler

DocBook er en meget komplisert standard som ikke nødvendigvis egner seg så godt for å lagre enkle bøker. En løsning på dette kunne vært å ha en fast kjerne som beskriver enkle bøker (se figur 6.2 på neste side). Her er kun de enkleste elementene som alle bøker består av representert. Bøker som kan lagres med denne kjernen kan for eksempel være glattsats (for eksempel romaner uten illustrasjoner).

Deretter kan man ha en tilleggsmodul som bygger på kjernen. Modulen kunne for eksempel inneholdt informasjon om illustrasjoner, tabeller, forskjellige kolonner og deres plassering. Den ville kunne dekke de fleste behov.

På toppen av denne kunne man bygget ut en modul som var spesifikk for de forskjellige typer bøker man ønsker å lage. Her er kun elementer som bare brukes i denne typen bok. Eksempler på dette er kokebøker, historiebøker, databøker, hestebøker og annet. I en kokebok vil man for eksempel kunne ha elementer som oppskrifter og ingredienser, hvis disse skal presenteres på en spesiell måte.

Ulempen med denne løsningen blir at det finnes veldig mange forskjellige moduler, og man må derfor passe på å inkludere de riktige modulene der disse brukes. Denne måten å løse DocBook-designet på



Figur 6.2: Forslag til ny struktur for DocBook

vil redusere kompleksiteten i enklere verk, der det ikke er nødvendig med elementer som inngår i spesielle bøker. Man vil stort sett få de elementene man trenger, og forhåpentligvis ikke så mye mer.

Kapittel 7

Metode og funn

I dette kapittelet ser vi på hvilken metode som er benyttet, og hvordan arbeidet med oppgaven ble gjennomført.

7.1 Kvalitativ analyse

I denne oppgaven har jeg gjort en kvalitativ analyse. Ifølge (Ratcliff 2004) er det tre forskjellige metoder å samle inn data på i kvalitativ forskning. Det ene er å gjøre semistrukturerte, åpne intervjuer. Det andre er å gjøre direkte observasjoner. Dette kan for eksempel være ved hjelp av videopptak, eller det kan være direkte observering. Den tredje metoden er å se på skrevne dokumenter som inneholder ord eller visuell data, og ikke tall.

Her har jeg til tider benyttet meg av alle disse metodene, på forskjellig måte. Jeg utførte to semistrukturerte intervjuer for å kartlegge den eksisterende manusflyten. Deretter analyserte jeg skrevne dokumenter om forskjellige filformater, og undersøkte artikler som omhandlet problemer som liknet på problemstillingen i denne oppgaven. Til sist observerte jeg presentasjoner av de to forskjellige datasystemene som var mulige løsninger. Flere detaljer om utførelsen av disse metodene er beskrevet i seksjon 7.3 på side 57. I hovedsak er det analyse av dokumenter jeg har benyttet i denne oppgaven.

Hele veien analyserte jeg de data jeg hadde samlet inn, sammenliknet dataene med hverandre. Deretter satte jeg dette opp mot problemstillingen.

7.1.1 Semistrukturerte intervjuer

Jeg hadde to semistrukturerte intervjuer med to personer fra redaksjonen norsk skjønnlitteratur. Begge personene hadde vært med på møtet der vi presenterte problemstillingen, så de kjente mer eller mindre

til denne. Det første intervjuet var med en produksjonskonsulent. Jeg valgte å intervjue henne fordi hun hadde en sentral rolle i Cappelens arbeidsflyt, og hun hadde mange forskjellige arbeidsoppgaver. Intervjuet er basert på min forståelse for systemet som ble forklart av IT-sjef Peter Hausken. Jeg stilte spørsmål som kunne hjelpe til med å kartlegge manusflyten. Jeg spurte henne også om hun hadde andre krav eller ønsker enn de som IT-sjefen hadde lagt fram.

Det andre intervjuet var med en datavant redaktør. Jeg valgte henne fordi redaktørene har en viktig rolle i manusflyten til forlaget. Hun hadde også eksperimentert litt med forskjellige Word-makroer, og var interessert i problemstillingen. Jeg hadde bedt henne om å se på figuren jeg hadde laget om Cappelens system på forhånd. Hun hadde gjort dette, og kom med synspunkter til den. Jeg stilte andre generelle spørsmål som jeg lurte på i forbindelse med systemet. I tillegg var jeg interessert i å vite mer om forlagets form for gjenbruk, og hvordan dette fungerte i dag. Til slutt stilte jeg spørsmål om hennes synspunkter til prosjektet vårt, om hun hadde tro på at dette ville hjelpe Cappelen.

7.1.2 Lesing av sekundærlitteratur

Den største delen av oppgaven dreide seg om sammenlikning og analyse av sekundærlitteratur. Det første jeg gjorde var å samle inn mengder av data. Jeg undersøkte hvilke formater som fantes, og samlet sammen informasjon om disse. Dette var både nettsider og manualer. Jeg valgte å bruke verdensveven mye ettersom denne inneholder enorme mengder informasjon, og det brukes en del forskjellige formater og standarder der. Blant annet XML og PDF er per dags dato mye utbredt for å dele informasjon på verdensveven.

Jeg plukket ut de viktigste egenskapene til hvert format, og satte de opp mot hverandre. I tillegg sammenliknet jeg formatenes egenskaper med kravene vi hadde kommet fram til for det nye systemet.

7.1.3 Etongrafisk forskning

Jeg har også benyttet metoden etnografisk forskning. Ifølge (Myers 2004) vil det si å undersøke den menneskelige, sosiale og organisasjonelle aspektene til utvikling av informasjonssystemer. Jeg har vært så heldig at jeg har kunnet arbeide med oppgaven min hos Cappelen en dag i uka det siste året. På den måten har jeg kunnet kommunisere med forlagets ansatte gjennom hele gjennomføringen av oppgaven. I tillegg har jeg vært med på en rekke møter sammen med personer fra forlaget og personer fra systemene vi evaluerte. I seksjon 7.3.8 på side 60 kan du se en oppsummering av møtene.

7.2 Utviklingsmodell

Jeg har kun tatt for meg den første delen i innføringen av et nytt datasystem. I denne oppgaven har jeg analysert behovene, utarbeidet en kravspesifikasjon, og kommet fram til et designforslag som kan tilfredstille denne. Det vil si at koding, testing og integrering ikke er dekket. I tillegg har jeg gått litt grundig inn på forskjellige formater og standarder som er aktuelle for forlagets bruk.

På analyse- og designbiten er det blitt brukt en lineær modell for å komme fram til resultatene. Det vil si at jeg først tok meg av analysedelen før jeg begynte med designet. Men innenfor de to blokkene har jeg arbeidet med forskjellige deler av systemet til samme tid, og ikke gått rett fra begynnelse til slutt, som i en evolusjonær modell.

Videre følger en mer detaljert beskrivelse av hva jeg har gjort.

7.3 Gjennomføring og funn

I denne seksjonen beskriver jeg hvordan jeg utførte de forskjellige delene av prosjektet.

I begynnelsen hadde jeg planer om å gjøre en undersøkelse blant forfatterne for å se om det var mulig å få de til å strukturere manusene sine selv. Men Cappelen ville at prosjektet i første omgang skulle holdes innenfor forlaget. Derfor gjennomførte jeg ikke dette.

7.3.1 Eksisterende system

Først hadde vi et møte internt i Cappelen. Der deltok forlagets IT-sjef, en produksjonskonsulent for oversatt skjønnlitteratur, en forlagssjef for oversatt skjønnlitteratur og en redaktør for generell litteratur. Her fikk vi representert flere redaksjoner, og personer med forskjellige oppgaver. IT-sjefen la fram sine ønsker til et nytt system for Cappelens elektroniske behandling av bøker. Vi fikk høre innspill fra de andre personene fra forlaget.

Forlagssjefen ble etter hvert sjef over en annen avdeling også, og fikk derfor mer ansvar og arbeid. Redaktøren fikk permisjon. Derfor var det produksjonskonsulenten jeg tok videre kontakt med. Jeg hadde et semistrukturert intervju med henne om arbeidsprosessene i forlaget, hvordan et manus behandles på de forskjellige stadiene i forlaget, hvilke andre parter som de samarbeider med, og hvilke oppgaver som hun hadde innenfor bedriften. Etter intervjuet kom jeg fram til et kart over manusflyten hos forlaget. Jeg sendte denne til henne. Hun leste igjennom min forklaring til figuren, og kom med flere synspunkter og pekte på deler jeg ikke hadde forstått bra nok.

Deretter hadde jeg et semistrukturert intervju med en redaktør for norsk skjønnlitteratur. I dette intervjuet fikk jeg svar på mange vesentlige spørsmål jeg hadde om Capelens system. Hun pekte på hva jeg ikke hadde forstått riktig, og informasjonen hun gav meg hjalp til med den endelige kartleggingen av det eksisterende systemet til forlaget. Hun kom med mye nyttig informasjon om både gjenbruk, og om sin egen plass i det nye systemet, og hvordan hun arbeidet i det nåværende systemet.

Den tekniske biten av systemet ble kartlagt ved å ta kontakt med IT-sjefen igjen, og jeg fikk et skriv om systemet SPARTA, som jeg tok utgangspunkt i under skrivingen.

7.3.2 Ønsket system

Det var IT-sjefen i Cappelen som presenterte sine krav og ønsker for hva forlaget trenger. Dette er noe forlaget har arbeidet med en stund, så problemstillingen ble nokså klar.

7.3.3 Sammenlikning

Jeg stykket opp manusflyten i deler, og sammenliknet hver del med ønskene og kravene som hadde blitt presentert. På denne måten fant jeg ut hva som kunne effektiviseres, og jeg fikk avdekket de kritiske stedene der problemene var vanskelige å løse. For å løse problemene lette jeg etter forskningsartikler som omhandlet liknende problemer. For det meste søkte jeg i ACM Digital Library(*ACM Digital Library* 2005) og med CiteSeer(*CiteSeer.IST - Scientific Literature Digital Library* 2005-)

7.3.4 Dokumentformater

Store deler av kapittelet om dokumentformater ble skrevet utifra informasjon hentet fra verdensveven. I tillegg til å lete på verdensveven hadde vi et møte med Håkon Wium Lie fra Opera Software. Han hadde skrevet en bok i CSS og HTML om CSS. Vi fikk høre om dette prosjektet, og hvilke synspunkter han hadde innenfor lagringsformater for bøker. Han mente at man fint kunne klare seg med å CSS og HTML når man skulle skrive en bok. Men han hadde ekstra programvare for å lage ting som for eksempel innholdsfortegnelse. Dette er noe man kunne laget automatisk ved hjelp av XSLT.

7.3.5 Standarder

Jeg undersøkte på verdensveven hvilke standarder som fantes når det gjalt lagring av bøker. Jeg lette etter DTD-er og XSD-er som kunne

passe best til problemstillingen. Jeg fant fram til en del DTD-er som jeg editerte og undersøkte. Ved å se hvilke elementer som var inneholdt i de forskjellige filene, og å lese om de forskjellige filene, fant vi fram til den DTD-en som vi syntes passet best for Cappelen.

Forenkling av DocBook

Det som var viktig med forenklingen av DocBook var å beholde strukturen, slik at dokumentene som er basert på denne ikke bryter med DocBook-standarden. For at det skulle bli lettere å arbeide med standarden valgte vi å bruke en XSD som var laget av DTD-en.

Det enkleste var å finne elementene som kun hørte hjemme i databøker. Dette gjaldt elementer som GUI-knapper, menyer, programlisting og så videre. DocBook inneholder en god del slike elementer, som opplagt er overflødige.

Deretter plukket jeg ut elementer som hadde med annet enn bøker å gjøre. Som for eksempel artikler. Vi trenger ikke kunne skrive en artikkel i vår DocBook-versjon. Men man bør likevel kunne referere til disse.

I tillegg finnes det mange funksjoner som er unødvendig detaljerte, som det ikke er så mye poeng å ta med. For eksempel finnes det et `callout`-element. Dette peker til et bestemt punkt i et bilde. Det er mange elementer som har med dette å gjøre, og vi valgte derfor å fjerne alle disse. Hvis det skulle være nyttig en gang, får man heller legge piler direkte inn i bildet selv.

Det neste som burde være greit å fjerne er elementer som omhandler konferanser. Her er det detaljer som sponsorer og deltakere som ikke har noe med en alminnelig bok å gjøre.

I tillegg har vi detaljert metadata som heller ikke har noe med boka å gjøre. Som for eksempel adressen og telefonnummeret til forfatteren. Slik informasjon bør heller lagres et eksternt sted der all informasjon om en forfatter lagres. Med dette mener jeg at det holder å skrive inn denne informasjonen i SPARTA (se seksjon 4.1.2 på side 23).

Slik gikk jeg fram og analyserte hvert element, inntil jeg satt med et subsett. Deretter ble en person fra Cappelen med meg og så over de elementene jeg var usikre på om forlaget hadde bruk for. Det var han som lagde den endelige versjonen som vi sendte av gårde til InfoFuture.

Forslag til endringer av DocBook

Vi oppdaget at det fantes visse svakheter ved DocBook-standarden. Ved å se på strukturen i andre formater kom vi opp med et forslag til en annen struktur for DocBook. Jeg har også fått noen idéer om DocBook fra møtene med InfoFuture.

7.3.6 Teknologi

Jeg tok for meg fire forskjellige XML-editorer for å se om dette var noe som kunne brukes for lite datavante personer. Jeg valgte fire kjente og mye brukte editorer som virket lovende. Jeg tok meg tid til å prøve dem, og teste ut forskjellige funksjoner editorene hadde å tilby. I tillegg sjekket jeg om de hadde støtte for validering mot en bestemt DTD. Spesielt la jeg vekt på brukergrensesnittet under testingen. Ettersom konklusjonen ble så opplagt, valgte jeg å ikke teste programmene på lite datavante personer, og heller ikke prøve flere av dem.

Før vi eventuelt skulle gå igang med å lage et nytt system, undersøkte vi om det vi trengte eksisterte fra før av. Vi fant to de to forskjellige firmaer som så ut til å kunne tilby systemer som kunne passe med problemstillingen. Firamet PDC-tangen kunne tilby oss systemet MOSES, og InfoFuture kunne tilby ContentMapper. IT-sjefen tok kontakt med disse firmaene. Begge var interessert i å levere sitt system til Cappelen.

Vi hadde flere møter med de to forskjellige firmaene, der vi presenterte Cappelens krav, og de kunne fortelle om de ville kunne oppnå disse kravene. I tillegg fikk vi presentasjoner, fremvisninger og demonstrasjoner av de to forskjellige systemene de kunne tilby forlaget.

Vi sammenliknet de to systemene, og så på dem i forhold til problemstillingen. I enda et møte på Cappelen ble forlaget enig om å prøve ett av disse systemene, det som virket mest lovende til å oppfylle de fleste betingelsene. Etter flere møter med det aktuelle firmaet, ble firmaet og forlaget enige om å sette igang et testprosjekt for å undersøke firmaets tilbud.

Det jeg har skrevet om MOSES og ContentMapper er kun basert på skriv og notater fra møtene og presentasjonene. Det vil si at vi ikke har testet programmene de har å tilby enda.

7.3.7 Forslag til ny manusflyt

Etter at problemet var analysert, og vi hadde kommet fram til hvilke verktøy og formater vi ønsket å benytte oss av, utarbeidet vi et forslag til ny manusflyt.

7.3.8 Oppsummering

I dette prosjektet har jeg hatt en rekke møter, noen intervjuer og skrevet noen e-poster. I tillegg har jeg gjort noen analyser og funn. Her er en oppsummering av disse:

Møter

- To møter med IT-sjef Peter Hausken for å kartlegge ønsket manusflyt
- To møter internt i Cappelen (ett før og ett for å diskutere tilbudene)
- Fem møter med InfoFuture. Vi diskuterte endringer av DocBook, innføring av systemet, samt gjennomføring av et testprosjekt
- Ett møte med PDC Tangen
- Tre møter med utvikler Paul Heisholt (to for omgjøring av DocBook, ett for diskusjon av Cappelens system)
- Ett møte med Håkon Wium Lie fra Opera Software for å høre om hans prosjekt der han skriver en bok i CSS og HTML om CSS

Intervjuer

- Ett semistrukturert intervju med produksjonskonsulent Randi Faye fra norsk skjønnlitteratur for å kartlegge eksisterende manusflyt
- Ett semistrukturert intervju med redaktøren Harriet Karoliussen fra norsk skjønnlitteratur for å kartlegge eksisterende manusflyt, og oppbygningen av bøker

Eposter

- Fire relevante e-poster med Peter Hausken (to for å kartlegge ønsket manusflyt, to for kartlegging av eksisterende manusflyt)
- Tre relevante e-poster sendt med produksjonskonsulenten Randi Faye for å kartlegge eksisterende manusflyt i etterkant av intervju

Analyser

- Sammenliknet ønsket system med eksisterende system
- Analyserte forskjellige formater og standarder sett i forhold til problemstillingen
- Evaluerte 4 forskjellige XML-editorer
- Sammenliknet systemene Contentmapper og MOSES

Funn

- Utarbeidet kart over det eksisterende systemet
- Kom frem til tre utfordringer for å effektivisere det eksisterende systemet
- Fant fram til hvilket format og hvilken standard som passer best for forlaget
- Kortet ned DocBook-standarden for Cappelens bruk
- Utarbeidet noen forslag til endringer av DocBook-standarden
- Fant ut hvilket system som passer best for forlaget
- Utarbeidet forslag til ny manusflyt

7.4 Videre arbeid

Planen for denne oppgaven var også å skrive om utførelsen av et testprosjekt av programvaren som vi diskuterte oss fram til. Men forlaget fikk mye å gjøre, og testprosjektet ble derfor utsatt kraftig. Jeg valgte å fortsette med det samme temaet uten å ta med testprosjektet, ettersom resultatene fra det ikke ville være klare før lenge etter at oppgaven min skulle være ferdig.

Kapittel 8

Strukturering

For å kunne konvertere et manus til XML må det struktureres først. Hvis man får manuset på en bestemt struktur kan man maskinelt konvertere det til XML. I et vanlig manus er det ingen ting som tilsier at en overskrift er en overskrift for datamaskinen. En datamaskin er ikke istand til å forstå semantikken til tekst i naturlig språk på samme måte som et menneske er. Vi er nødt til å fortelle datamaskinen eksplisitt hvilke forskjellige deler en tekst består av.

Hvis alle forfattere skrev alle overskrifter med uthevet skrift i størrelse 16, hadde det ikke vært noe problem. Da kunne vi fortelle datamaskinen at dette er tegnet på at en tekst er en overskrift. Men slik fungerer ikke verden. Noen ganger er det bare en linje uten punktum som er en overskrift, og andre ganger kan det skrives på helt andre måter. Vi er nødt til å standardisere måten et manus blir skrevet på, slik at vi lett kan skrive algoritmer til datamaskinen for at den skal forstå semantikken. Men hvordan kan vi få manus over på riktig strukturert from? Og hvem er det som bør gjøre dette?

8.1 Strukturering av forfattere og oversettere

Redaksjonen har liten tro på at man kan få forfattere til å strukturere sine dokumenter selv. Det fører til en del ekstra arbeid, som forfatterne

Figur 8.1: Fra ustrukturert dokument til XML

sannsynligvis ikke er interessert i. Tanken på riktig struktur kan også stjele oppmerksomhet fra innholdet, og man kan få et kvalitetstap. All erfaring tilsier at man ikke kan forvente at forfatterne vil klare eller være villige til å strukturere selv. Folk som har hatt mye med forfattere å gjøre, tror de fleste at forfatterne ikke en gang vil kunne følge en enkel Word-mal. Det er derfor veldig optimistisk å tenke seg at forfattere kan produsere noe som direkte lar seg konvertere til XML. Men det finnes alltid unntak, og kanskje man kan få enkelte til å følge en struktur. For eksempel oversettere. For å ikke få misfornøyde forfattere bør de oppmuntres til å bruke stilen, og ikke tvinges. Selv om det viste seg at forfatterne klarte å strukturere delvis hadde ikke dette hjulpet. For å kunne konvertere til XML kreves det 100% riktig strukturering. Men la oss se på hvilke alternativer vi har når vi snakker om at forfatterne skal strukturere sitt eget manus.

Det første alternativet er å gi skribentene en XML-editor. I seksjonen 9.2 på side 69 om XML-editorer er det beskrevet en nøye gjennomgang og evaluering av fire forskjellige XML-editorer, for å undersøke om de er enkle nok til at en lite datavant person kan benytte seg av dem. Vi kan regne med at forfattere og oversettere ikke vil sette seg ned og lære XML for å bruke en XML-editor mens de skriver.

Det neste alternativet er å gi manusskriverne en Word-mal som inneholder de riktige DocBook-elementene, som lar seg konvertere til XML. Dette utelukker de få prosentene som ikke leverer i Word-formatet. Uansett hvilken løsning vi velger vil noen falle bort, og vi må finne løsninger for disse situasjonene også. Word-malen må være lett å følge hvis man velger dette alternativet. Problemet med en slik mal er at det er lett å avvike fra den. Og da er man kommet nesten like kort som uten strukturering. I dette tilfellet må man uansett gå inn og redigere manuset manuelt før det lar seg konvertere til XML. Dette vil kun være en løsning hvis man får skriverne til å følge malen fullt ut.

En måte å lage en Word-mal på er å bruke fargekoder. Først skriver man en tekstbit, deretter velger man fra en liste hva slags element denne biten er. Når alt er dekket med farger er manuset ferdigstrukturert. Hvis man gjør dette holder man likevel skriveren borte fra design og grafisk utforming, og fokuserer på innhold. Dette fortsetter at man har stilløse alternativer som gir farger uten designmessig mening. Men her er det en viss risiko for at skriveren kan bli forvirret av alle fargene.

En siste måte å få forfatterne til å strukturere selv, er å gi dem et program som tar seg av struktureringen. Det vil si et interaktivt program som kun aksepterer å skrive riktig strukturerte dokumenter, der ulovlige ting ikke lar seg gjøre. Det bør også dukke opp advarsler hvis skriveren avviker fra den angitte malen. For at personer som ikke nødvendigvis er datavante skal kunne bruke dette programmet bør det være grafisk, enkelt og intuitivt. En grei måte er at det minner om

Microsofts programmer, som er kjent av mange brukere. Hvis et slikt program ville la seg oppdrive, hadde dette sannsynligvis vært en god løsning.

8.2 Strukturering av redaksjonen

Det som er en mer realistisk løsning er å la redaksjonen i forlaget ta seg av struktureringen. Slik det gjøres i dag, der alt skjer manuelt, er nok ikke den beste løsningen. Man er nødt til å komme fram til en enklere måte å strukturere på, som er tidsbesparende. Det går an å leie inn ekstra folk som tar seg av dette.

Det går an å få redaksjonen til å ta i bruk en XML-editor for strukturering av manus. Men da er man avhengig av å ha personer med XML-kompetanse og forståelse for Docbook XSD-en. Dette stiller større krav til de ansatte, og en opplæringsprosess må gjennomføres. Men det største problemet med denne løsningen er at den sannsynligvis vil være like lite effektiv som dagens prosess. Å tillegge et hvert element et tilsvarende XML-element vil være bortkastet tid å gjøre fullstendig manuelt. Spesielt ettersom et godt strukturert dokument som følger en standard (f.eks Docbook XSD-en) er nok til å konvertere til XML.

En annen aktuell løsning kan være at en redaktør kun redigerer programmet strukturelt, i henhold til en standard. Resultatet av denne operasjonen blir deretter konvertert til XML. Men også på denne måten vil det skje unødvendig mye manuelt, og det burde være mulig å spare inn litt tid her ved å gjøre noe automatisk.

8.3 Automatisk strukturering

Vi har sett på hvordan strukturering kan gjøres av forfattere, oversettere og av redaksjonen. I den påfølgende teksten skal vi se på muligheten for å gjøre en automatisk strukturering. Det er sannsynligvis ikke mulig å la en datamaskin ta seg av all strukturering for alle bøker. Det er mange elementer som en datamaskin ikke kan styre. En maskin kan ikke se om en bok er vakker, eller om den er god å lese. Heller ikke kan en datamaskin plassere bilder på en side på en pedagogisk korrekt måte. Mange forfattere har sin egen måte å skrive sine bøker på. Et struktureringsprogram må altså kunne gjette seg fram til strukturen i et manus bare ved å se på formateringen. Dette kan bli en veldig stor oppgave å lage, ettersom det finnes uendelig med forskjellige elementer som skal gjenkjennes.

Å lage et program som gjenkjenner enkelte vanlige elementer burde være mulig. For eksempel bør det gå an å kjenne igjen et avsnitt.

Og korte linjer med større skriftstørrelse eller fet skrifttype kunne gjenkjennes som en overskrift. Hvis datamaskinen skal gjette seg til hvilke elementer som har hvilken struktur, vil man alltid måtte ha en manuell gjennomgang og kontroll til slutt. Det kan hende enkelte ting er tvetydige, og datamaskinen kan gjette feil.

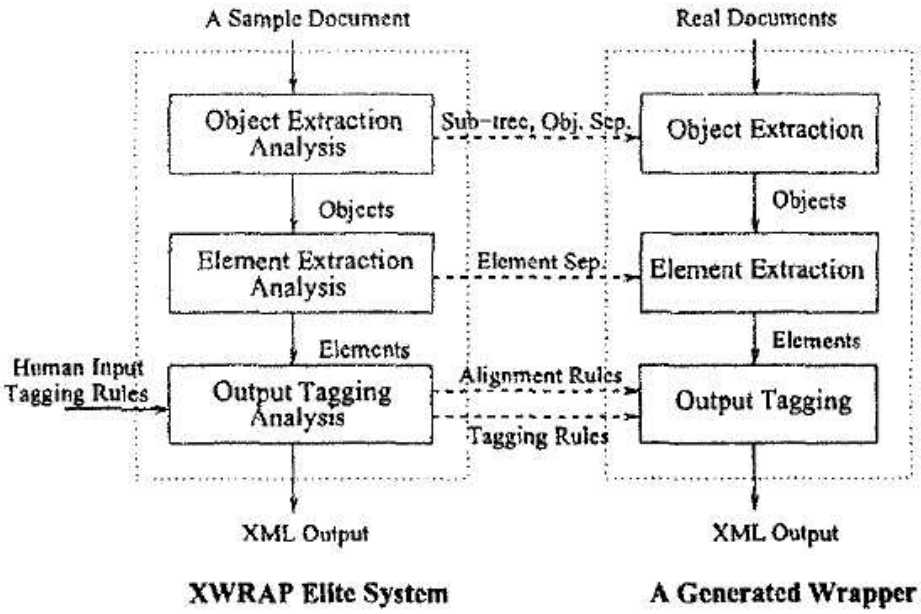
En måte å lage et slikt gjenkjenningsprogram kan være å bruke kunstig intelligens. Man kan lage et program som lærer og utvikler seg. Ved at man mater maskinen med mange forskjellige overskrifter, og forteller den at de er overskrifter. Etter hvert når man får inn et manus som ikke følger en struktur programmet har lært, lærer den seg det. Slik blir programmet bedre og bedre.

I artikkelen "A Trainable System for the Extraction of Meaning From Text"(Bagga 1995) drøftes en måte å trekke meningfull tekst ut fra forskjellige tekster. Systemet deler opp teksten basert på en ordbok som inneholder omkring 150000 ord. Systemet hjelper brukeren med å trekke ut en semantisk nettverksrepresentasjon fra treningsartikler som er lagret i en omfattende database. Basert på brukerens trening former systemet statistiske tabeller, en kunnskapsbase, og et sett med regler som speiler brukerens aksjoner blir utformet. Systemet generaliserer reglene. Ved å bruke en semantisk klassifikasjon som er basert på statistikk legger systemet disse reglene til nye artikler i databasen, og bygger automatisk nye semantiske nettverk. Artikkelen forklarer i detalj hvordan dette ble utført.

8.3.1 XWRAP Elite

"Wrapping Web Data into XML"(Han 2001) er også en artikkel som omhandler automatisk strukturering. I denne artikkelen er det brukt såkalte ’wrappers’ for å gjøre om fra HTML til XML slik at programvare kan forstå strukturen i dokumentene. Utviklingen av slike wrappers er en sakte og omfattende prosess. Derfor ble XWRAP Elite utviklet som et verktøy for å lage wrappers. Det første den gjør er å finne ut hvor data er lokalisert i HTML-dokumentet, og dele data inn i individuelle objekter. Deretter deles disse opp i dataelementer. Til slutt markeres objektene og elementene i et strukturert format. På denne måten forkortes interaksjonen med mennesker.

I HTML har man tagger som angir overskrifter etc. Dette finnes ikke i vår situasjon. Utover dette gjenkjenner wrappere laget med XWRAP for lite til at det kan brukes av oss. Det står også skrevet i artikkelen at det er en utfordring å forminske den menneskelige interaksjonen ytterligere, slik at mest mulig går automatisk. I hoveddelen i artikkelen presenteres det algoritmer og heuristikker som forminsker mengden av semantisk input som skal til for å trekke ut informasjon. Prosessen der wrappers blir generert beskrives også i artikkelen.



Figur 8.2: XWRAP Elite

8.4 Konklusjon

Det greieste alternativet er å ha en sammensetning av både manuell og automatisk gjenkjenning. Maskinen kan gjenkjenne de enkle elementene, og et menneske tar seg av resten. En enkel roman vil sannsynligvis kunne gjenkjennes på en grei måte. Men derimot en grunnskolebok med flere bilder, kolonner, punktlister, sitater og annet på hver side vil det bli et større manuelt arbeid med.

Vi må i tillegg finne et program som kan konvertere fra et ferdig strukturert dokument til XML. Programmet må kunne validere mot DocBook XSD-en.

Og deretter trenger vi programvare for å konvertere til PDF. Denne programvaren må kunne brekke om bøker, og få det til å bli en bok ut av manuskriptet. På litt avanserte bøker skal designere gå inn og lage stilark (CSS) til de forskjellige dokumentene.

Vi trenger også programvare for å lagre de ferdige XML- og PDF-filene i databasen. Sparta har et grensesnitt der dette vil la seg gjøre. Men det enkleste hadde vært å ha programmer som direkte kan laste opp og hente fra den. Dette programmet må kunne lagre bilder og tekst side om side.

Vi må ha programvare som kan hente ut manuskripter fra databasen,

og legge på eventuelle stilark. Hvis man er interessert i å publisere manuskripter på nettet eller i annen elektronisk form, må vi også få tak i programmer for dette. Den siste PDF-en som lagres i systemet må være behandlet manuelt. Hvis ikke kan man risikere at bøkene ikke er pent ombrukket, og ikke pedagogisk korrekte for bøker som må være det. Redaktører stiller høye krav for hvor god leseligheten av en bok må være. En datamaskin forstår ikke hva som er behagelig å lese for et menneske, så vi må derfor ha et menneske til å gjøre finpussen på manuset i slutten av prosessen.

Når den siste PDF-en er ferdig lagres denne, og sendes til trykkeriet. Her er det også mulig å generere en JDF og sende med.

Kapittel 9

Teknologi

9.1 Krav til verktøy

Vi har funnet ut at Cappelen ønsker å benytte XML. Da gjenstår det å finne gode XML-verktøyer. Det viktigste med dette verktøyet er at det er brukervennlig på en slik måte at lite datavante personer kan bruke det. Det er ønskelig med et program der man ikke trenger å ha mye kunnskaper om XML og DTD/XSD for å skrive et validerende manus. Programmet må kunne validere bøkene mot DocBook XSD-en. Det er en fordel at programmet kan eksportere XML-filene til PDF. Ellers må vi finne ekstra verktøy som kan gjøre dette.

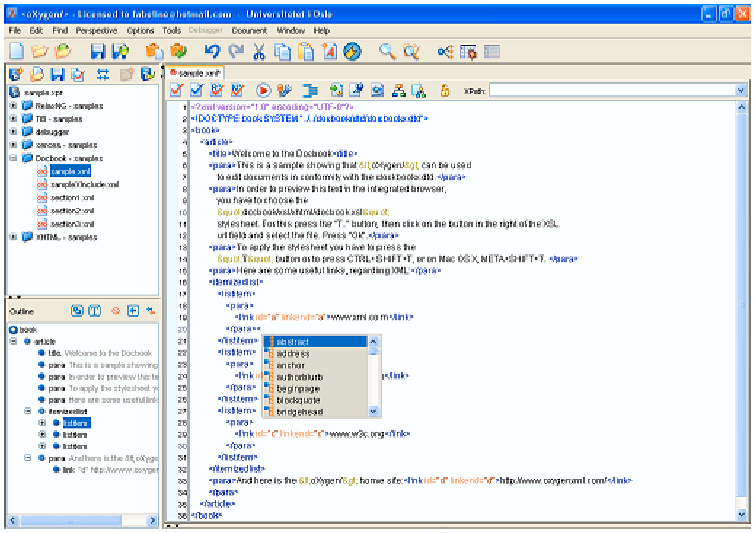
9.2 XML-tekstbehandlere

En mulighet er å bruke XML-tekstbehandlere. Jeg har sett på fire forskjellige tekstbehandlere for å se om dette er noe som kan brukes. Jeg har lagt mest vekt på grensesnittet mot brukeren.

9.2.1 oXygen XML editor 4.1

Oxygen er en tekstbasert XML-tekstbehandler. Man jobber direkte med koden i dokumentet, men får stor hjelp av tekstbehandleren. Når man begynner med en ny tagg (skriver '<') dukker det opp en boks der man kan velge mellom lovlige elmenter. Når man har valgt et element kommer dette til syne, inkludert elementer som må være inneholdt i dette.

Programmet er veldig intuitivt. Er man vant til tekstbehandlere forstår man lett hvordan dette skal brukes. Ettersom tekstbehandleren er kodeorientert kan man anta at folk som ikke er vant til kode kan ha problemer med å bruke den. Programmet passer best for folk som allerede har en viss insikt i XML. Det kan hende folk synes det ser rotete ut at alle tagger er synlige.



Figur 9.1: Skjermbilde av oXygen XML editor

Valideringen skjer ikke interaktivt, bare når man trykker på en valideringsknapp. Ettersom man får mye hjelp underveis ser ikke dette ut til å være noe problem. Noe som kan være litt slitsomt med tekstbehandleren er når man skal starte et nytt element. Man må trykke seg fram til slutten av elementet man holder på med før man kan starte på et nytt.

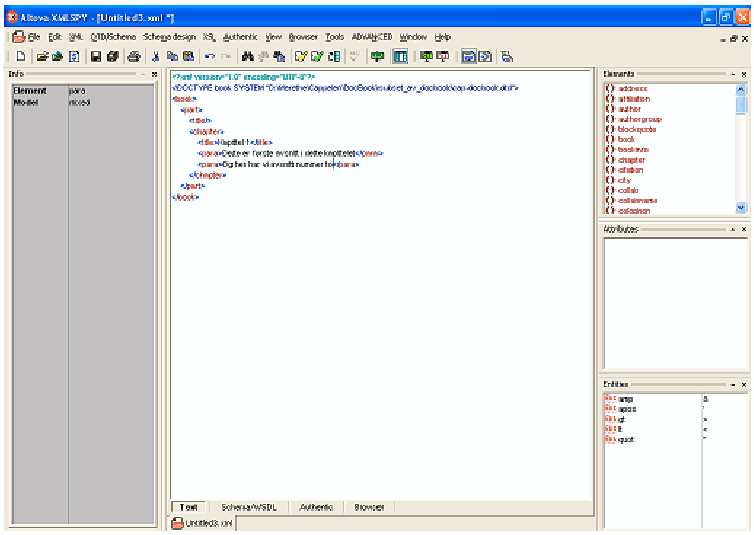
Ellers er det veldig lett å åpne den tilhørende DTD-en/XSD-en hvis man er i tvil om reglene. Til venstre har man et lite kart over strukturen. Og det er lett å skifte mellom åpne dokumenter. Det er i tillegg mulig å lagre flere filer i et prosjekt som lett kan aksesseres i en meny til venstre.

Det følger med en 'Tree editor' der man kan redigere XML-treet. Her kan man skifte ut elementer, og se informasjon og regler for de forskjellige delene.

Evaluering

Tekstbehandleren er tekstbasert og kodeorientert. Dette medfører at det lett kan brukes av datavante, men folk som har lite kunnskap om XML og DocBook kan ha problemer. Derfor egner det seg ikke veldig for forfattere, ettersom vi antar disse har lite datakunnskaper.

Man kan benytte dette programmet for å gjøre om dokumentets form fra ustrukturert til strukturert. Dette gjør man ved å lage et nytt dokument med riktig DTD/XSD, lime inn det som skal, og deretter sette inn riktige tagger. Men dette er ingen god løsning ettersom den medfører mye arbeid.



Figur 9.2: Skjermbilde av XMLSPY 2004

Programmet kan ikke eksportere filer til PDF. Hvis dette programmet skal benyttes er man avhengig av å ha en ekstra applikasjon til som kan ta seg av dette.

9.2.2 XMLSPY 2004 Home Edition

Jeg har evaluert “home edition”-versjonen av XMLSPY som er gratis. XMLSPY er tekst- og kodebasert, og den minner den veldig om Oxygene. Tekstbehandlerene har lik måter å behandle tekst på, og de har hjelpebokser som dukker opp når man begynner å skrive inn en ny tagg. I XMLSPY starter man med et tomt skjelett som følger den angite DTD-en eller XSD-en når man skal begynne et nytt dokument. Dette gjør at man får litt hjelp til hvordan man skal fortsette å skrive.

Skjermbildet er delt i tre. Til venstre får man informasjon om elementet markørern er innenfor. I det midtre vinduet er selve koden. Til høyre kan man velge neste element, attributt eller entitet. Disse hjelpevinduene kan lett slås av. Programmet søtter funksjoner som vanlige tekstbehandlere pleier å tilby. I tillegg har man en ’pretty-print’-funksjon som rydder opp i koden, og får den til å se pen ut.

I ekspertversjonen kan man i tillegg eksportere til og importere fra en database. Man kan også eksportere og importere tekst. I tillegg får man med en mer avansert tekstfremviser.

Evaluering

XMLSPY er en behagelig tekstbehandler som passer best for kodevante personer. Den egner seg derfor ikke spesielt til bruk for forfattere og oversettere. Men programmet kan brukes av datavante medhjelpere i Cappelen for å få et dokument over på strukturert form, selv om dette ikke vil være en god løsning ettersom det medfører mye arbeid.

I hovedversjonen av programmet har man mulighet for å lagre direkte til databasen. Dette kan gjøre manusarbeidet enklere. Men programmet støtter ikke eksportering til PDF, så det trengs i tillegg verktøy for dette.

9.2.3 XMetaL Author

XMeTaL er en WYSIWYG[1] XML-tekstbehandler. Man trenger XMetaL Central for å kunne behandle DTD-er og XSD-er. Jeg testet Author-versjonen ettersom man kunne laste ned en gratis 30-dagers evalueringsversjon av denne.

Man kan velge mellom å starte et nytt XML-dokument eller et nytt velformet XML-dokument. Skjermen er tredelt. I det midtre vinduet er selve dokumentet. Til venstre ser man strukturen. Her kan man klikke på de forskjellige overskriftene for å gå til bestemte steder i dokumentet. Til høyre er en liste over elementer man kan sette inn.

Ved å dobbeltklikke på 'insert element' kan man legge til nye elementer. Deretter kan man velge riktig element, og det blir plassert i dokumentet. Så kan man behandle elementet videre. Jeg synes programmet til tider var litt uoversiktelig, og jeg var ikke alltid sikker på hvilket element jeg arbeidet med.

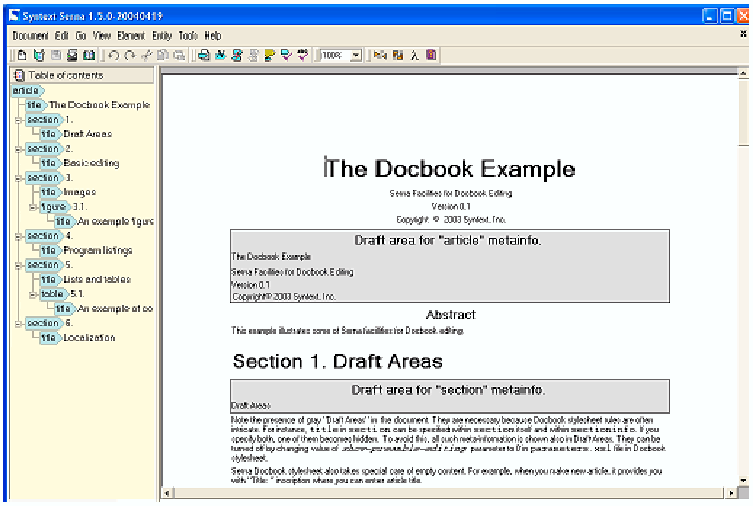
Evaluering

Ettersom tekstbehandleren er WYSIWYG kan den brukes av personer som ikke er veldig datavante, men noe kunnskap må man likevel ha om Docbook. Det som er positivt med tekstbehandleren er at man kan lagre direkte til PDF og HTML. I tillegg kan programmet laste direkte inn fra Word.

9.2.4 Syntext Serna

Serna er en grafisk XML-tekstbehandler. Den har oversiktelig og greit brukergrensesnitt. Men ikke alt er like intuitivt, så man trenger god opplæring for å ta programmet i bruk. Man kan velge hvilke nye

[1] 'What you see is what you get'. Det vil si at redigeringen ser ut slik som det ferdige resultatet blir.



Figur 9.3: Skjermbilde av Syntext Serna

elementer man skal sette inn fra en liste, eller skrive inn selv. Det er lagt inn snarveier for å gjøre enkelte operasjoner. Det vil si at man klarer å skrive et gyldig DocBook-dokument kun ved bruk av tastaturet, noe som gjør det effektivt å bruke denne tekstbehandleren selv om den er grafisk. Det er lagt inn enkelte funksjoner som gjør skrivingen mer behagelig. For eksempel kan man sette inn et nytt element likt det forrige hvis man trykker på linjeskift.

Selve skrivingen er interaktiv, og man får feilmeldinger på en statusbar nederst hvis man gjør feil i forhold til den angitte DTD-en/XSD-en, og feilen som blir gjort rettes opp. Til venstre har man et oversiktstre over strukturen til dokumentet. Dette gjør det enkelt å navigere og å holde oversikten. Det er også enkelt å skifte mellom forskjellige åpne filer.

Det finnes to forskjellige modi tekstbehandleren kan stilles i. Den første er WYSIWYG-modus der det du skriver ser litt ut som slik det kommer til å bli. Den andre modusen er tagg-modus der du ser hvilke elementer de forskjellige delene er. Det finnes sikkert flere funksjoner i programmet som ikke har blitt testet ut.

Programmet kan åpne filer i ren tekst. Hvis man åpner et dokument som ikke følger en bestent DTD/XSD (for eksempel en vanlig tekstfil) går programmet i en tekstmodus som skrur av alle XML-funksjoner. Tekstbehandleren kan eksportere dokumenter til PDF.

Evaluering

Man kan ikke åpne et ustrukturert dokument for så å legge til XML-elementer, for da går programmet over i tekstmodus der XML-funksjoner er avslått. Hvis man prøver å kopiere en tekst for så å lime det inn i dokumentet forsvinner alle linjeskift. Derfor ser jeg ingen god måte å benytte dette programmet i prosessen med å gå over fra et ustrukturert til strukturert dokument.

Men tekstbehandleren kan benyttes av forfattere og oversettere som har noe greie på XML og Docbook. Når man har satt seg litt inn i programmet går skriveprosessen effektivt ved hjelp av tastesnarveier. Men det kan by på problemer hvis forfatteren har lite kunnskap om XML og DocBook. Ettersom DocBook er veldig omfattende er menyene lange, og det kan være vanskelig å vite hvilke elementer man skal bruke.

9.2.5 Konklusjon

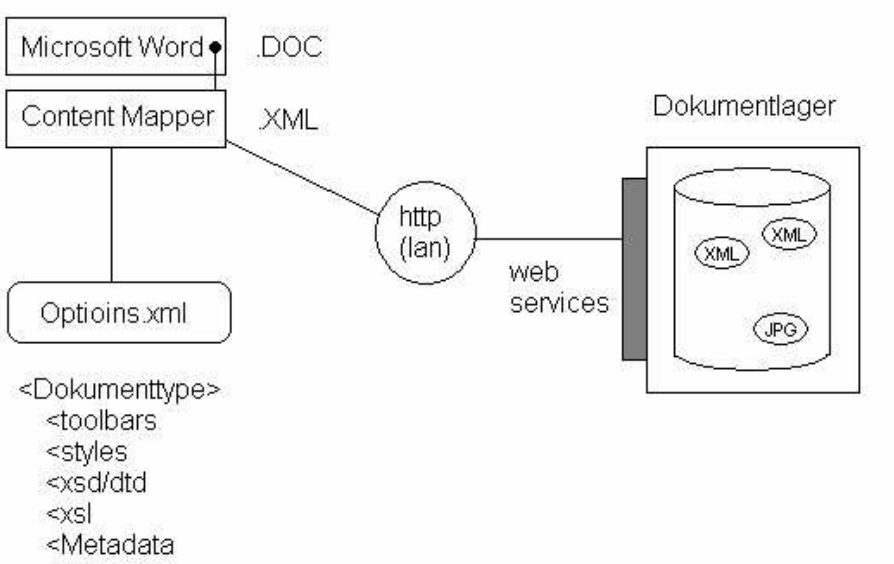
Det vil bli veldig mye ekstraarbeid hvis manuskripter skal skrives i XML-tekstbehandlere. Det vil også bli mye arbeid å bruke en XML-tekstbehandler til å gjøre om fra et ustrukturert til strukturert dokument. Ingen av tekstbehandlerene vi har sett på har et enkelt nok brukergrensesnitt til at personer som ikke vet noe om XML eller DocBook kan bruke det. Men det kan være en idé å bruke en av XML-tekstbehandlerene for spesielt bruk. Hvis det er bruk for at teknisk personell skal kunne gå inn og manuelt endre små detaljer.

9.3 Annen programvare

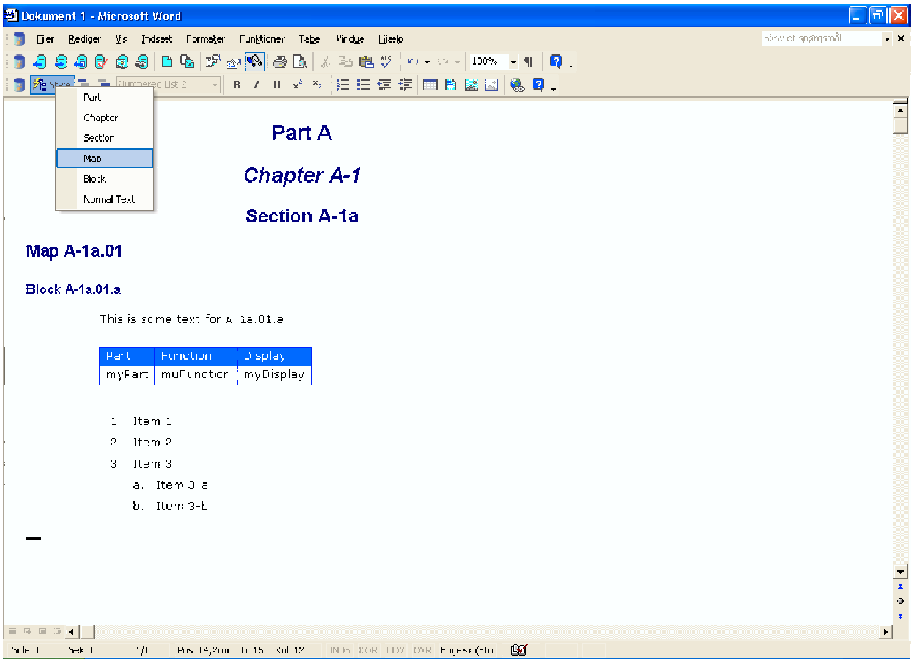
Vi har tatt kontakt med to forskjellige systemutviklingsfirmaer som har utviklet hvert sitt system, og disse har gitt oss to forskjellige tilbud som kan være aktuelle for forlaget. I denne seksjonen diskuterer vi oss frem til hvilket tilbud som best dekker forlagets ønsker.

9.3.1 Content Mapper

Content Mapper er et verktøy som er utviklet av det lille, danske firamet InfoFuture. Produktet er nyutviklet. Content Mapper er bygget på tekstbehandleren Microsoft Word, og kan eksportere et dokument til en rekke forskjellige formater som for eksempel XML, DOC og PDF. Tanken bak systemet er at det skal være enkelt å generere et XML-dokument som følger en bestemt DTD eller XSD uten at brukeren trenger å ha kunnskap om XML.



Figur 9.4: Kart over Content Mapper



Figur 9.5: Skjermbilde av Content Mapper

Content Mapper er utvidelsen til Word som gjør seg synlig i nye knapper og menyvalg. Figur 9.4 på forrige side[2] er en oversikt over strukturen til Content Mapper, og figur 9.5 på forrige side[3] viser et skjermbilde av programmet. De fleste forfattere som skriver for maskin leverer manus skrevet i Word. Derfor er det et stort pluss at systemet er en utvidelse av dette programmet. Når man lager et nytt dokument i Content Mapper sørger systemet for at det som skrives er i samsvar med den predefinerte DTD-en/XSD-en. Hvis reglene brytes får brukeren gode feilmeldinger som leder vedkommede i riktig retning. Det er lagt inn automatikk som gjør at nye overskrifter og nye blokker automatisk får riktig stil.

De vanligste elementene i et dokument kan implementeres med Content Mapper. Det er innebygde funksjoner for å lage overskrifter, underoverskrifter, avsnitt, kulelister, numererte lister, innsetting av bilder og tabeller. Det er også mulighet for å referere til andre dokumenter i hvert enkelt dokument.

Programmet aksesserer en egen XML-fil med definerte behov ved konvertering, lagring og fremvisning. Denne filen er lett tilgjengelig og redigérbar ved hjelp av andre funksjoner. Her lagres hvilke knapper og menyer du ønsker å tilføre din Word-kopi, hvilke stiler du ønsker på dokumentene, hvilke regler som skal brukes (DTD/XSD), hvilken XSL/XSLT som skal brukes, og hvilke metadata som skal lagres for dokumentet.

Med dette programmet har man mulighet til å eksportere sitt strukturerte dokument til forskjellige formater. Blant annet XML, HTML og PDF. Content Mapper kan også hente inn filer fra forskjellige formater. For at disse skal kunne eksporteres er man avhengig av at de følger strukturen som er definert i konfigurasjonsfilen. Hvis denne ikke er fulgt får brukeren veiledende feilmeldinger som gjør det greit å strukturere dokumentet.

Filene som genereres kan enten lagres lokalt på maskinen som brukes, eller de kan eksporters til en ekstern database. Grensesnittet mellom programmet og serveren er integrert i Content Mapper. Data blir overført via http ved hjelp av web services som snakker med databasen.

I databasen lagres tekst og bilder side om side. Det vil si at det kun lagres referanser til bildene i dokumentene, og bildene lagres eksternt. Dette gir muligheter for gjenbruk og er plassbesparende. Bildene lagres både i full størrelse og skalert slik de skal vises på trykk.

[2]Figuren tilhører InfoFuture.
[3]Figuren tilhører InfoFuture.

Konklusjon

Med Content Mapper løses ikke problemet med å komme fra et ustruktuert til et strukturert dokument. Utviklerne av Content Mapper mener det vil være en nærmest uløselig oppgave. Men Content Mapper kan tilby en hjelp til å gjøre dette. Den manuelle arbeidsprosessen vil derfor kunne effektiviseres ved bruk av Content Mapper. Det gjenstår å finne en god rutine for å gjøre dette.

Hvis forfattere eller oversettere er villig til å ta i bruk Content Mapper har vi løst en viktig del av problemstillingen. Dette vil spare Cappelen for verdifull tid. Med dette systemet er det ingen sak å få et strukturert dokument konvertert til nyttig XML. Den neste biten som består i å lagre et XML-dokument i Cappelens forlagsystem, Sparta, vil også være løst med noen finjusteringer av Content Mapper-systemet. Infofuture er igang med å skreddersy programmet sitt mot Cappelens versjon av DocBook hvis forlaget ønsker det.

Men InfoFuture er et lite firma, og det kan derfor være mer risikabelt å bruke deres løsning. Det kan være tryggere å heller benytte systemene til store, kjente firmaer.

9.3.2 MOSES

MOSES er et annet system som firmaet PDC-Tangen tilbyr Cappelen. Dette systemet baserer seg på at forfatteren eller redaksjonen strukturerer manus i henhold til en Word-mal. Denne strukturerte filen blir deretter sendt inn til systemet MOSES, som lagrer den som en XML-fil. Ved å endre litt i sitt system, kan de få til å bruke DocBook DTD-en/XSD-en som Cappelen ønsker. Fra MOSES kan man hente ut XML-filen, eller eksportere den til designformater som InDesign, Quark og Framemaker. Foreløpig bruker systemet Framemaker for ombrekking. For enkle bøker kan dette skje automatisk, men med godkjenning av et menneske etterpå. Når boka er ferdig ombrukket er det administrasjonen i PDC-Tangen som retter opp feil. De var også under utvikling av en grafisk XML-tekstbehandler for å rette opp feil direkte i XML-filen. Firmaet kan tenke seg at man vil kunne automatisere 1/3 av ombrekkningen, 1/3 kan gjøres mer effektivt (med for eksempel bedre koder fra MOSES), og at 1/3 må produseres som før. Med dette mener utviklerne av MOSES at Cappelen kan spare store kostnader.

Designet vil her skje etter at dokumentet når MOSES, dvs at Cappelen ikke har like god kontroll som om dette skulle vært gjort internt i huset. I tillegg må Cappelen betale per manus som går gjennom MOSES-systemet. Det ble påpekt at de ikke ble låst til å velge bestemte trykkerier. I tillegg kan de tilby bruk av JDF, som forteller om hvordan en bok er oppbygd med tanke på trykkeriet. Det er ikke alle trykkerier

som bruker dette enda, men det er på vei inn ifølge møterepresentanter fra PDC-Tangen.

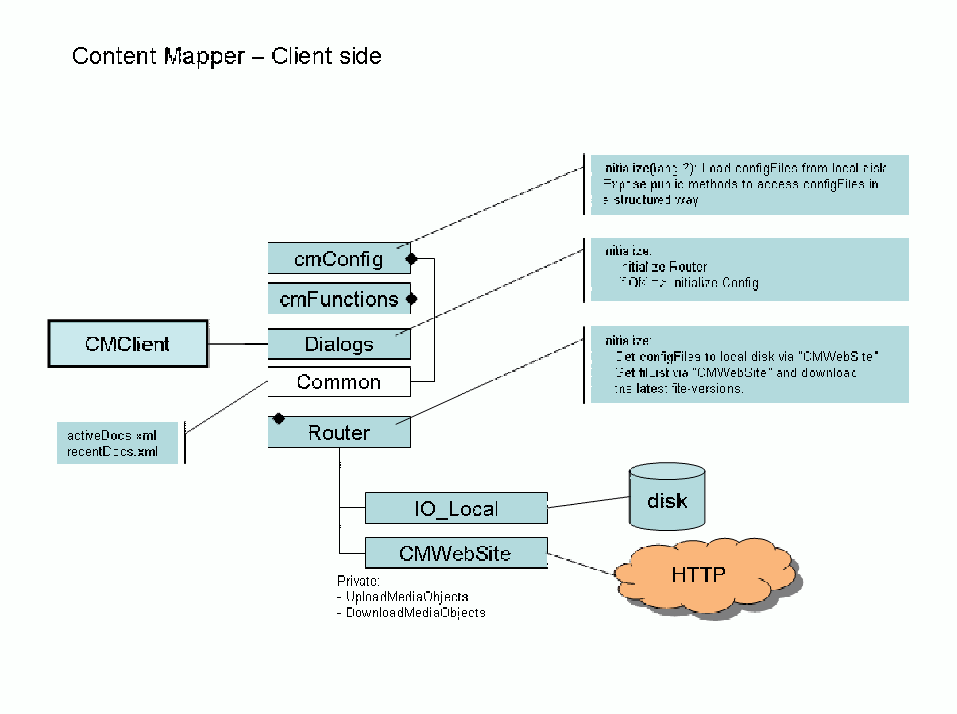
Konklusjon

Hvis Cappelen velger å bruke MOSES overfører de en del kontroll av manus til PDC Tangen. De betaler for hvert manus som går igjennom deres system. Dette medfører mindre arbeid for forlaget, men de mister en del kontroll. I tillegg vil de knytte seg til, og bli avhengig av PDC Tangen. Denne løsningen løser ingen del av problemet med å komme fra et ustrukturert til et strukturert dokument.

9.4 Konklusjon

- Begge tilbudene har mulighet for å eksportere både til PDF og til XML.
- Systemet MOSES går utifra at man har klart å strukturere sitt dokument i henhold til en word-mal. Dette kan bli vanskelig å følge for forfattere. Det er lett å avvike fra malen. Når man skriver sitt manus i Content Mapper vil man hele tiden få beskjed om hva som avviker fra reglene, og hjelp til å løse dette.
- MOSES bruker ingen åpen standard i sin XML-struktur for bøker, men en standard som de har utviklet selv. Cappelen er skeptisk for å låse seg til denne.
- Med Content Mapper vil mer av manusprosessen komme inn i forlaget, og de vil trenge flere interne ressurser. Men med MOSES vil forlaget gi fra seg en del av denne prosessen til et eksternt firma.
- Hvis MOSES benyttes vil Cappelen bli avhengig av PDC-Tangen. Men med Content Mapper vil alt foregå internt i firmaet, og forlaget vil få mer frihet. Selv om de kanskje vil være litt avhengig av support fra InfoFuture i begynnelsen. Dette systemet kjøpes, og vil bli forlagets eiendom.
- Ingen av tilbudene løser hele problemet med å komme fra et ustrukturert til et strukturert dokument. MOSES har ingen løsning. Men Content Mapper kan gjøre noe av arbeidet. Det kan strukturere det den forstår utifra sammenhengen, og gi meldinger om hva som ikke ble strukturert. Dette kan brukes for å effektivisere denne prosessen.

Utifra listen over ser det ut til at det er Content Mapper som best oppfyller Cappelens krav og ønsker.



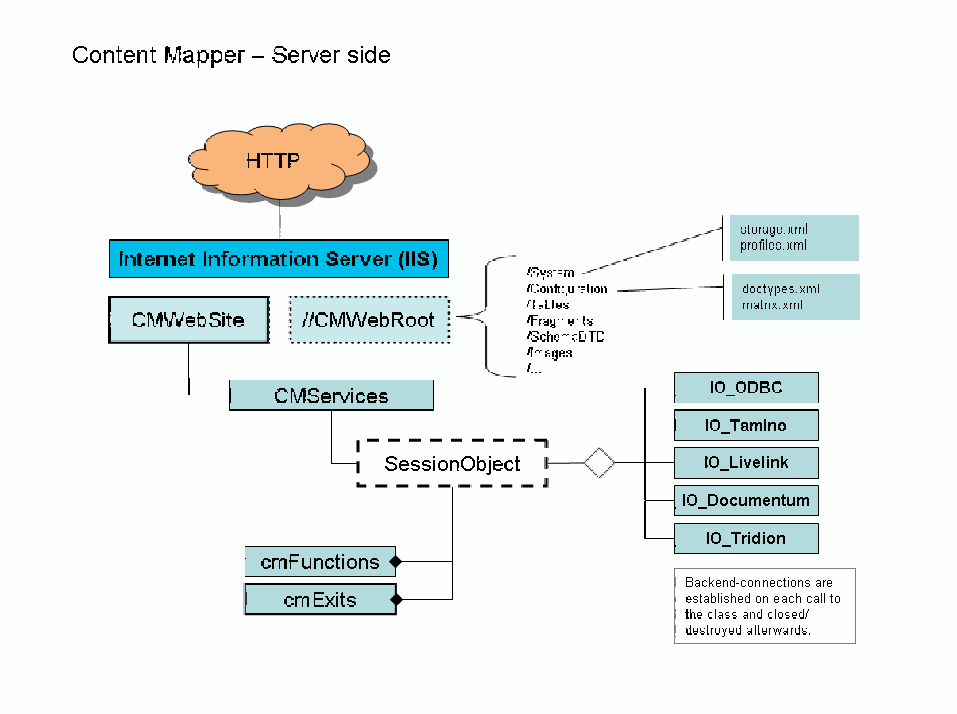
Figur 9.6: Kart over klientsiden av Content Mapper

9.5 Content Mapper

Vi velger å se mer på Content Mapper som forlagssystem i denne oppgaven. I denne seksjonen ser vi litt mer på verktøyet, og ser hvordan det kan integreres med resten av Cappelens system.

9.5.1 Teknisk

Content Mapper er bygget på toppen av MS Office og MS Word. I tillegg trenger man .NET Framework 1.1. Programmet er skrevet i C#.NET, VB.NET (Visual basic) og J#.NET (Java-sharp). XML og XSLT er også brukt, og det benytter XML-nettjenester (eng. web services). All kommunikasjon mellom motorene og mellom motor og brukergrensesnittet bli gjort via XML. Det betyr at programmet lett kan konfigureres (i forskjellige XML-filer), og at det er lett å utvide applikasjonen.



Figur 9.7: Kart over tjenestesiden av Content Mapper

Klient

Systemet består av klientmaskiner som kommuniserer med en tjener (se figur 9.6 på side 79[4]). Content Mapper-klienten har sine egne konfigurasjonsfiler. Den har blant annet en ruter-funksjon. Gjennom denne kan man enten aksessere det lokale lageret, eller kommunisere med tjeneren over HTTP. Informasjon overføres som mediaobjekter.

Tjener

Klientmaskinene har kontakt med en felles internettinformasjonstjener (se figur 9.7 på forrige side[5]). De kommuniserer gjennom Content Mapper sin nettside, via nettjenester. Tjeneren har et sesjonsobjekt som tar seg av alle funksjoner. Blant annet inn-ut-operasjoner. Tjeneren har mulighet for å kommunisere med databaseaksessystemet ODBC, XML-tjeneren Tamino, Livelink Enterprise Server, Documentums Digital Asset Management og Tridion Global Content Management. (*Content Mapper - An InfoFuture A/S White Paper* 2004)

9.5.2 Funksjoner

I denne seksjonen ser vi på noen av funksjonene som Content Mapper tilbyr. Blant annet hvordan brukergresesnittet er, hvilke formater den kan operer med og hvordan den kommuniserer med innholdsstyrings-systemet.

Brukergrensesnitt

Brukergrensesnittet er basert på MS Word. I tillegg er den en del forenklet, 70-80% av alle Wordfunksjoner og knapper er gjemt. Funksjonene som er relevante for denne typen publisering er ikke gjemt dypt ned i menyer, men lett tilgjengelig. Alle knapper og menyer kan konfigureres via konfigurasjonsfiler som er lagret i XML.

Content Mapper støtter flere forskjellige maler der man kun fyller inn blanke felter. Kompliserte tabeller kan også limes rett inn. Det er også lagt inn en hjelpefunksjon som kan hjelpe til med å lage nye dokumenter og å sette disse opp riktig.

Det er også mulig å forhåndsvise ditt dokument uten lagring først, ved å bruke integrerte eller egendefinerte XSLT-stilark.

[4]Figuren tilhører InfoFuture.
[5]Figuren tilhører InfoFuture.

Dokumentformater

Content Mapper kan konvertere til blant annet Word, XML, HTML, og PDF. Teoretisk sett vil det gå an å eksportere et strukturert format til hvilket som helst annet tekstformat. Konverteringene gjøres via XML og XSLT-stilark. I konverteringer mellom Word og XML brukes vektorbasert Word-grafikk (VML). Programmet støtter validering mot XSD-er eller DTD-er. I tillegg støtter den kryptert XML-innhold. Ettersom XML skal skrives med tegnsettet UTF-8 har programmet naturligvis støtte for dette.

Fra et ustrukturert format til et som er strukturert går ikke konverteringen helautomatisk. En intelligent funksjon prøver å strukturere innholdet, og brukeren blir gjort oppmerksom på hva som ikke blir plassert i forhold til den angitte DTD-en eller XSD-en. Dette betyr at problemet med å komme fra ustrukturert til strukturert ikke er løst fullstendig. Mye kan konverteres automatisk, mens resten må det manuell hjelp til med. På denne måten kan man effektivisere prosessen, selv om den ikke blir helautomatisert.

Innholdsstyringssystem

Programmet opererer med direkte tilgang til innholdstyringsystemet (CMS = content management system). Det jobber med mange forskjellige CMS-er når man skal åpne, lagre, sette inn og gjenbruke innholdskomponenter. Man kan både legge til og hente ut komponenter, bilder og hyperlinker fra CMS-en.

Det er også integrert et frittstående XML-basert lager som man kan jobbe med når man ikke har kontakt med databasen, eller hvis man vil ha en egen, lokal kopi.(*Content Mapper - An InfoFuture A/S White Paper* 2004)

9.5.3 Content Mapper hos Cappelen

InfoFuture er villige til å tilpasse Content Mapper til å lage dokumenter som følger Cappelens versjon av DocBook. Det vil si at de kommer til å lage ekstra knapper og menyvalg som gjør det enkelt å sette inn elementer som beskrives i denne XSD-en. De vil gjerne ha én XSD som de kan implementere i programmet sitt. De er også villige til å lage forskjellige maler til de tre forskjellige typene bøker Cappelen har kommet fram til. Disse malene følger forskjellige subset av Cappelens versjon av DocBook. Det vil si en mal for de enkleste strukturerte bøkene (for eksempel romaner), en mal for bøker som i tillegg inneholder tabeller og illustrasjoner, og en siste mal som dekker de fleste andre

behov for mer avanserte bøker. Knappene som er synlige avhenger av hvilken mal man har valgt.

Per i dag foregår et testprosjekt hos Cappelen for å teste ut at Content Mapper fungerer tilfredstillende, og for å sjekke om dette er et interessant verktøy for forlaget.

Kapittel 10

Konklusjon og oppsummering

Som en avslutning for oppgaven presenterer vi et forslag til en ny manusflyt hos Cappelen, basert på hvilke verktøy og formater vi anbefaler Cappelen å bruke.

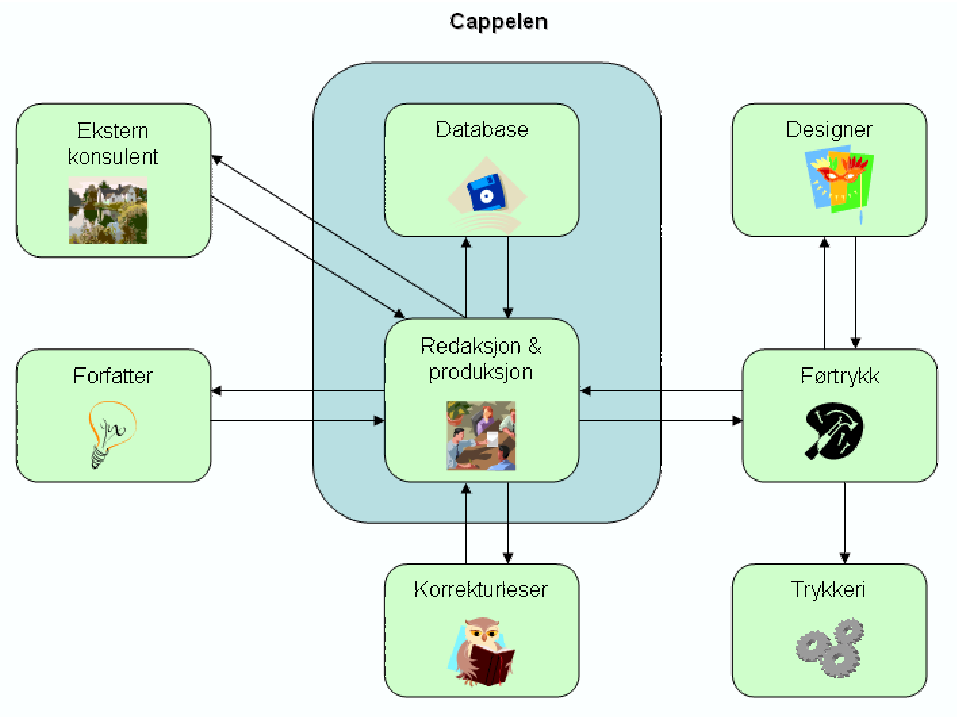
10.1 Forslag til ny manusflyt

Kort oppsummert utløper den nåværende manusflyten seg slik; Et ustrukturert dokument kommer inn fra forfatteren. Deretter blir det manuelt designet, og det konverteres til PDF. Denne sendes til trykkeriet, og lagres i databasen. Designet og ombrekkingen gjøres som regel utenfor forlaget.

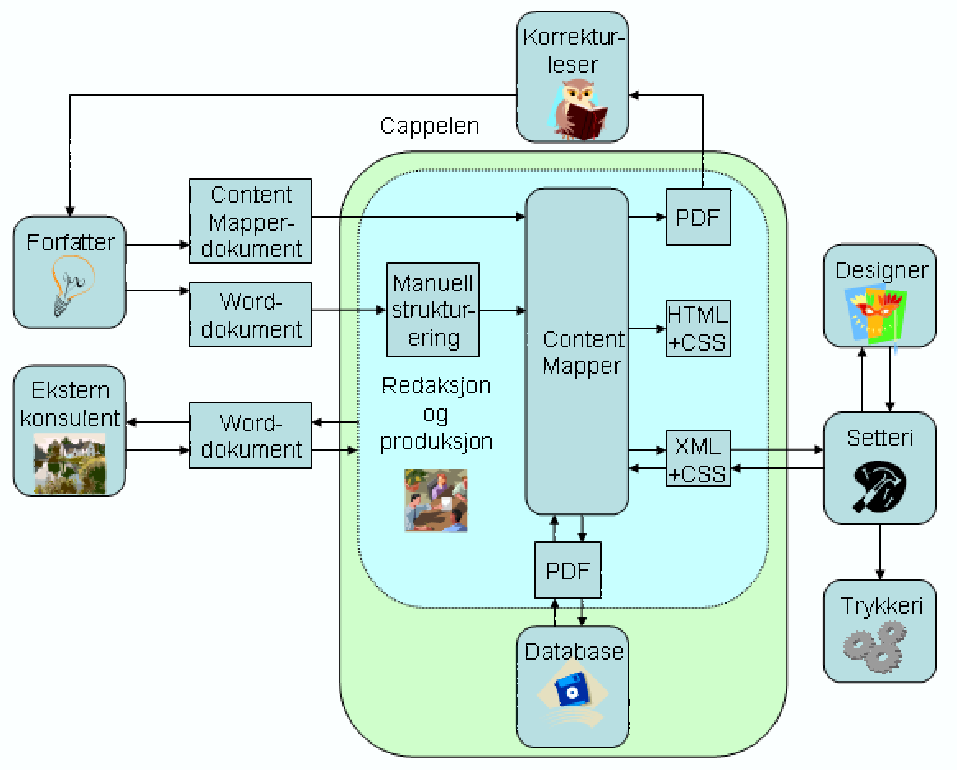
I denne seksjonen blir det presentert et forslag til en ny manusflyt basert på resultatene som er kommet fram gjennom oppgaven. Kapitelet kan tolkes som et sammendrag av disse. Figur 10.1 på neste side er et kart over et forslag til en ny manusflyt, og figur 10.2 på side 86 er en mer detaljert oversikt. Jeg har kommet fram til disse to figurene ved å sette sammen resultatene fra resten av oppgaven. Vesentlig informasjon som ikke blir presentert på figuren vil forklares i teksten.

1. Det finnes to forskjellige begynnelser på den nye manusflyten. Den første er at en forfatter leverer et ustrukturert dokument inn til forlaget. Dette manuset blir bearbeidet manuelt av Cappelens ansatte slik at det blir riktig strukturert, og kan behandles av Content Mapper.

   Den andre måten er at en forfatter eller oversetter arbeider direkte i Content Mapper mens hun/han skriver. På denne måten struktureres manuset mens forfatteren arbeider med det. Ved å gjøre dette slipper man å ha en manuell strukturering i ettertid. Vi antar at det kun er et fåtall av oversetterne og forfatterne som er villige til å gjøre dette, eller klarer å få det til.



Figur 10.1: Forslag til ny manusflyt (oversikt)



Figur 10.2: Forslag til ny manusflyt (detaljer)

2. Content Mapper konverterer deretter det ferdigstrukturerte manuset til XML, og validerer det mot Cappelens versjon av DocBook XSD-en. Hvis struktureringen ikke var vellykket kommer programmet med feilmeldinger og tilbyr en hjelp til å rette opp feilene.

3. Nå har vi fått et gyldig DocBook-dokument som lagres direkte i Sparta via Content Mappers lagringsmekanisme.

4. Bilder lagres eksternt i databasen. Kun en referanse til disse lagres i manuset på XML-form. Bildene lagres både i full størrelse, og i den skalerte størrelsen som skal brukes i manuset.

5. Etter dette legges ferdiglagde stilark til manuset. Vi benytter XSLT til å plukke ut vesentlig informasjon fra XML-dokumentet, og vi bruker XSL og/eller CSS for å presentere det.

6. Hele manuset med tilsatte stilark blir deretter konvertert til andre lesbare dokumenter. Men det som produseres her er kun godt nok for gjennomlesning og korrektur, og ikke for trykk. Dette er fordi den er automatisk generert. 1.korrektur genereres her som en PDF-fil.

7. Denne PDF-filen blir lest igjennom, og ønskelige rettelser blir påpekt. Deretter sendes dette til forfatteren.

8. Hvis forfatteren leverte et ustrukturert dokument får vedkommede nå en ferdigstrukturert fil i ønsket format av sitt manus. Personen retter deretter opp feil. Så må manuset gjennom en manuell strukturering, og deretter genereres det en XML via Content Mapper igjen. Hvis forfatteren har strukturert manuset sitt selv kan personen rette opp feilene direkte i Content Mapper. Det er allerede definert stilark til dokumentet, og man kan derfor jobbe direkte i XML-filen.

9. Ettersom stilark allerede er definert kan man konvertere dokumentet rett til en PDF igjen, for at manuset skal kunne gjennomgå 2.korrektur.

10. Deretter leses PDF-filen igjennom igjen, og feil som er igjen blir påpekt. Etter dette sendes filen til forfatteren.

11. På samme måte som sist (pkt 8) vil enten forfatteren få en strukturert fil som rettes opp, og konverteres til XML igjen. Eller så vil personen få tilbake XML-filen slik at feil kan rettes opp med Content Mapper. Igjen har vi en XML-fil med tilhørende stilark. Forlaget antar at denne er tilnærmet feilfri.

12. Så har tiden kommet for å legge til det manuelle designet. Dette gjøres som oftest av eksterne designere. Boka blir ombrukket på en behagelig og lesbar måte, illustrasjoner blir lagt til, og andre ting blir gjort pent. Dette kan ikke gjøres automatisk, ettersom det er vanskelig å fortelle datamaskinen hva som er pent eller pedagogisk.

13. Når designet er på plass blir det generert en 'preflight'-PDF. Etter at denne er godkjent lagres den i databasen, både som PDF, XML og de tilhørende stilarkene.

14. En JDF lages av redaksjonen og legges ved. Det ferdige resultatet (JDF og PDF) blir sendt til trykkeriet.

10.2 Videre arbeid

Det neste steget er å fullføre testprosjektet hos Cappelen i samarbeid med InfoFuture. Dette prosjektet ble startet i april 2005. Denne testingen vil bli gjort som en undersøkelse for å finne ut om programvaren kan tilby det forlaget ønsker, og om resultatene fra denne oppgaven er tilfredstillende. Vi skal teste ut den nye manusflyten der bruk av Content Mapper og Cappelens versjon av DocBook står i fokus.

Det vil bli lagt mest vekt på brukeropplevelsen og nytten av systemet. Det vil også bli testet ut om Content Mapper sin konvertering til PDF gir dokumenter som er greie for korrekturlesning. Testingen vil i første omgang kun foregå innad i forlaget, og ikke i samarbeid med forfattere. Det er planlagt at en evaluering skal være ferdig til høsten. Hvis Cappelen blir fornøyd med det nye systemet vil de gjennomføre et mer omfattende prosjekt. Hvis dette også viser seg å bli vellykket vil systemet bli tatt i bruk i de deler av virksomheten det virker hensiktsmessig.

Bibliografi

ACM Digital Library (2005),
Nettside: http://citeseer.ist.psu.edu/cs. The Association for Computing Machinery. Besøkt: 2005-01-07.

Adobe PDF (2004),
Nettside: http://www.adobe.com/products/acrobat/adobepdf.html. Besøkt: 2004-10-25.

An Introduction to the Text Encoding Initiative (TEI), DTD (2001),
Nettside: http://gutenberg.hwg.org/teidtds.html. HTML Writers Guild, Inc. Besøkt: 2005-01-19.

Artesia TEAMS (2004),
Nettside: http://www.artesia.com/html/teams.html. Besøkt: 2004-12-16.

Bagga, A. (1995), 'A trainable system for the extraction of meaning from text', *IBM Press* .

Book DTD's I (2001),
Nettside: http://gutenberg.hwg.org/gutdtds1.html. HTML Writers Guild, Inc. Besøkt: 2005-01-19.

Bray, T. (2004), 'Extensible markup language (xml) 1.0 (third edition)',
Nettside: http://www.w3.org/TR/REC-xml/. W3C. Besøkt: 2004-10-18.

Brockmeier, J. (2001), *DocBook Publisihing*, Prima Tech.

CiteSeer.IST - Scientific Literature Digital Library (2005-),
Nettside: http://citeseer.ist.psu.edu/cs. Penn State's School of Information Sciences and Technology. Besøkt: 2005-01-07.

Content Mapper - An InfoFuture A/S White Paper (2004), Tilsendt.

Dataspråk (2004),
Nettside: http://www.cip4.org/intro.html#What_Is_JDF. Fraunhofer Institute for Computer Graphics. Besøkt: 2004-11-25.

Dataspråk (2004-),
Nettside: http://www.sprakrad.no/templates/Page.aspx?id=270. Besøkt: 2004-11-18.

Flynn, P. (2003), 'The xml faq',
Nettside: http://www.ucc.ie/xml/. UCC. Besøkt: 2004-04-15.

Garshol, L. M. (1999), 'En introduksjon til xml',
Nettside: http://www.garshol.priv.no/download/text/xml-intro/. Besøkt: 2004-10-17.

Gudgin, M. (2001), 'Xml 1.0 and namespaces',
Nettside: http://www.awprofessional.com/articles/article.asp?p=24502&seqNum=6. Addison-Wesley. Besøkt: 2004-10-20.

Han, W. (2001), 'Wrapping web data into xml', *ACM SIGMOD Record, Volume 30 , Issue 3* .

Hannemyr, G. (2003), 'Utkast til lov om åpne digitale formater og tjenester',
Nettside: http://folk.uio.no/gisle/essay/lovof.html. Gisle Hannemyr. Besøkt: 2004-04-19.

JDF White Paper (2005-),
Nettside: http://www.cip4.org/documents/jdf_overview/jdf_wp.pdf. CIP4. Besøkt: 2005-01-26.

J.W. Cappelens Forlag AS 175 år i 2004 (2004),
Nettside: http://www.cappelen.no/main/info.aspx?f=2270&artid=3271. OpenOffice.org. Besøkt: 2004-05-21.

Langmyhr, D. (1997), 'Nytt program for produksjon av mindre dokumenter',
Nettside: http://www.ifi.uio.no/in165/startex.html. Universitetet i Oslo. Besøkt: 2005-01-27.

Legal Notice (2005),
Nettside: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odcXMLRef/html/odcXMLRefLegalNotice.asp. Microsoft. Besøkt: 2005-02-28.

Myers, M. D. (2004), 'Ethnographic research in information systems',
Nettside: http://ww2.cis.temple.edu/isworld/vmc/aispert/myers.html. University of Auckland. Besøkt: 2005-04-15.

Office 2003 XML Reference Schemas Overview (2005),
Nettside: http://www.microsoft.com/office/xml/overview.mspx. Microsoft. Besøkt: 2005-02-28.

Open Document Format for Office Applications (OpenDocument) 1.0, commitee draft 2 (2004),
Nettside: http://www.oasis-open.org/committees/download.php/10765/office-spec-1.0-cd-2.pdf. OASIS. Besøkt: 2005-02-28.

PDF Reference, fifth edition (2005-),
Nettside: http://partners.adobe.com/public/developer/pdf/index_reference.html#5. Adobe Systems Incorporated. Besøkt: 2005-01-27.

Pepper, S. (2001), 'Emnekart og norsk elektronisk innhold',
Nettside: http://www.ontopia.net/topicmaps/materials/enorge.html. Ontopia. Besøkt: 2004-03-30.

Ratcliff, D. (2004), 'Qualitative research methods',
Nettside: http://www.vanguard.edu/faculty/dratcliff/index.cfm?doc_id=4254. Vanguard University. Besøkt: 2005-02-02.

Rees, I. C. (2004-), 'What is xml?',
Nettside: http://www.geocities.com/SiliconValley/Peaks/5957/wxml.html. Besøkt: 2004-10-17.

Rich Text Format (RTF) Version 1.5 Specification (2004-),
Nettside: http://www.biblioscape.com/rtf15_spec.htm. Besøkt: 2004-10-17.

Sparta - A book publishing management system (2004).

Treese, W. (1998), 'Putting it together: what's all the noise about xml?',
netWorker archive, Volume 2, Issue 5 .

Walsh, N. (2004), 'Validity',
Nettside: http://www.xml.com/pub/a/98/10/guide3.html. O'Reilly Media, Inc. Besøkt: 2004-10-17.

XML (2004),
Nettside: http://www.idealliance.org/standards_xml.asp. Besøkt: 2004-10-17.

XML 1.0 and Namespaces (2001),
Nettside: http://www.w3schools.com/xml/xml_namespaces.asp. Besøkt: 2004-10-20.

XML Attributes (2004),
Nettside: http://www.w3schools.com/xml/xml_attributes.asp. Besøkt: 2004-10-17.

Tillegg A

Oversettelser

Liste over norske uttrykk som er brukt. Noe er fra norsk språkråd(*Dataspråk* 2004-) (markeres med *), andre har jeg oversatt selv eller plukket opp tilfeldig. Jeg har ikke oversatt egennavn (f.eks. Extensible Stylesheet Language) eller forkortelser (f.eks. XML).

- frittstående - standalone
- godseid - proprietary
- grafisk utforming - layout
- navneområder - name spaces
- nettjenester - web services
- hjelpefunksjon - wizard
- informasjonshåndteren - management information services
- innholdsstrøm - content stream
- innholdstyringsystem - content management system
- internettinformasjonstjener - internet information server
- kontorverktøy - office tools
- markeringsspråk - markup language
- ruter - router
- skanne - scan
- skjermbilde - screenshot
- tagg - tag*

- verdensveven - world wide web*
- lenke - link
- syntaksanalyserer - parser

Tillegg B

Forkortelser

- CMS - Content Managmenent System
- CSS - Cascading Style Sheets:
- DAM - Digital Asset Management
- DB - Database
- DTD - Document Type Definition
- GPL - General Public License
- HTML - HyperText Markup Language
- HTTP - HyperText Transfer Protocol
- IIS - Internet Information Server
- JDF - Job Definition Format
- MS - Microsoft
- OASIS - Organization for the Advancement of Structured Information Standards
- ODBC - Open DataBase Connectivity
- PDF - Portable Document Format
- RTF - Rich Text Format
- SGML - Standard Generalized Markup Language
- TIFF - Tag Image File Format
- VML - Vector Markup Language

- WYSIWYG - What you see is what you get
- XML - Extensible Markup Language
- XSD - XML Schema Definition
- XSL - Extensible Stylesheet Language
- XSLT - Extensible Stylesheet Language Transformation

Tillegg C

Intervjuskjema 1

Intervjuskjemaet er basert på første utkast av eksisterende manusflyt, som er skrevet utifra Hauskens tegning og forklaring.

1. Forfatteren skriver et manus:
Hvordan jobber forfatteren?
Hvilke programmer bruker de?
Hvordan er manuset oppbygd?
Kunne forfatteren tenke seg å få ekstra tillegg for mer struktureringsarbeid? F.eks ved å følge en Word-mal?
Har forfatteren kompetanse til å gjøre dette?

2. Manuset blir tatt imot på Cappelen:
Hvem tar i mot?
Hvordan blir det sendt?
Hvilket format

3. Manuset blir tatt hånd om:
Hvordan lagres så manuset?
Hvem gjør dette?
Leses det igjennom for å se om det er interessant?
Hvem gjør isåfall dette?
Hvordan skjer det?

4. Manuset blir manuelt redigert:
Hvem gjør dette?
Hvordan skjer det?
Hvilken programvare brukes?

5. Manuset blir designet:

Hvem designer?
Hvem bestemmer hva som skjer?
Hvordan skjer dette?

6. Manuset blir sendt til trykkeriet:
Hvordan?
Hvordan ser dokumentet ut når det blir sendt?
Hvilket format?
Hvor mye er gjort av korrektur etc?

7. Manuset lagres hos Cappelen:
Hvordan blir det tatt imot?
Hvordan lagres det til slutt?

8. Spørsmål til hver person:
Hva er deres arbeidsoppgaver?
Hvordan gjør de disse?
Har personene involvert ønsker til forbedre system?
Fungerer alt som det skal?
Hva er problematisk med eksisterende system?
Er personene fornøyde?
Kunne de tenke seg en enklere måte å gjøre ting på?

Tillegg D

Intervjuskjema 2

Intervju med datavant redaktør.
Har bedt henne om å se på kapitelet om manusflyt og oppygningen av en roman.

Manusflyt
---
* Hvordan skjer antakelse av manus? Hvem er involvert?
* Hvordan fungerer konsulentuttalelsene? Hvor passer det inn i manusflyten?
* Hva er produksjonens rolle? Hvor kommer den inn i bildet?
* Er det med et administrasjonselement, eller kommuniserer parter direkte med redaksjonen?

* Er manusflyten annerledes når det gjelder utvikling av andre bøker enn romaner?

* Forklar hva du gjør fra du får inn et manus. Hvilke programmer brukes?

* Hvordan skjer ombrekkingen? Automatisk? Isåfall hvilke verktøy?
* Hvaslags programmer brukes det fra manuset blir innsendt til det er ferdig designet? Designverktøy?
* Hvilke program bruker setteriene?

Gjenbruk
---
* Hvordan gjenbrukes manus i dag? Hvordan kommer man fra PDF-filen til noe rettbart?
* Er det ofte manus gjenbrukes? Hvilke manus gjenbrukes?

Oppbygningen av en roman
----
* Flere innspill på oppbygningen av en roman?

Semantisk format
----
* Tror du forfattere kan klare å strukturere selv?
* Tror du at du hadde arbeidet mer effektivt med CM?

* Flere innspill/bekymringer fra redaksjonen?